Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Mes Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-17

Tipo de artículo: Artículo originales
Temática: Programación paralela y distribuida
Recibido: 19/04/2022 | Aceptado: 14/06/2022 | Publicado: 07/07/2022

Herramienta para el cálculo de la centralidad en redes multicapas

A new framework for centrality on multilayer networks

Armando Díaz Matos 0000-0002-4946-1639[1*]
Darian Horacio Grass Boada 0000-0002-9865-1615[2]

[1]Centro de Desarrollo de Aplicaciones, Tecnologías y Sistemas (DATYS). s/n, Avenida Patricio Lumumba, Santiago de Cuba, Cuba. armando.diaz@datys.cu
[2]Centro de Aplicaciones de Tecnologías de Avanzada. 7ma A #21406 E/ 214 y 216, Rpto. Siboney, Playa, La Habana, Cuba. horacio.boada09@gmail.com

[*]Autor para correspondencia: (armando.diaz@datys.cu)

RESUMEN

El análisis de la relevancia de las entidades que conforman una red es una tarea que permite analizar, comprender y hasta predecir el futuro de los miembros de la red así como el flujo y alcance de la información. En las redes donde existen varios tipos de relaciones entre sus entidades se denominan redes multicapas, ejemplo de ellas son las redes sociales donde usuarios conectan con distintos tipos de parentescos ya sean familiares, laborales o etc. En este tipo de redes, el cálculo de la relevancia o centralidad de las entidades es una tarea retadora debido a las múltiples forma de analizar este concepto y al costo computacional que acarrea su cálculo en redes de grandes dimensiones.

En este trabajo se presenta una herramienta para el cálculo de las medidas de centralidad en redes multicapas. Para su elaboración se empleó el lenguaje Scala y el *framework* Spark, con el propósito de calcular de forma paralela y distribuida las diferentes métricas de centralidad. En el análisis de la relevancia se consideran los diferentes niveles semánticos determinados por una selección de los tipos de aristas, así como su mezcla.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Mes Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-17

Los experimentos realizados alcanzaron niveles de aceleración de hasta 28.65 y 25.48 veces para la medida de Cercanía e Intermediación respectivamente. De igual forma, pero en menor medida, la métrica de Vector Propio aceleró 6.91 veces. Los resultados demuestran que las medidas de centralidad implementadas escalan tanto vertical como horizontalmente.

Palabras clave: Redes Multicapas, Medidas de Centralidad, Spark Framework.

ABSTRACT

The analysis of the relevance of the entities that make up a network is a task that allows to analyze, understand and even predict the future of the members of the network as well as the flow and scope of the information. Are called multilayer networks those that have several types of relationships between their entities, an example of which are social networks where users connect with different types of relationships, whether they are family, work or etc. Calculating the relevance or centrality of entities in multilayer networks is a challenging task due to the multiple ways of analyzing this concept and the computational cost involved in calculating it in large networks.

This paper presents a tool for calculating centrality measures in multilayer networks. It used Scala language and the Spark framework with the purpose of calculating in a parallel and distributed way the different centrality metrics. In the analysis of relevance, the different semantic levels determined by a selection of the types of edges as well as their merge are considered. The experiments speed up to 28.65 and 25.48 times for the measure of Closeness and Intermediation, respectively. Similarly, but to a lesser extent, the Eigenvector metrics reached accelerations of 6.91 times. The results show that the measures of centrality implemented scale.

Keywords: *Multilayer Networks, Centrality Measures, Spark Framework*

Introducción

Las redes son estructuras frecuentes que describen fenómenos relacionales de la naturaleza tales como las actividades sociales humanas, las comunicaciones, sistemas computacionales y redes biológicas por tan solo mencionar algunos. En estos sistemas, la interacción entre los componentes constituyen la red, la cual puede

denominarse red de información sin pérdida de generalidad. Cuando la información que se maneja por las entidades o sus relaciones presentan diferencias en su tipo se puede afirmar que es una red de información heterogénea (Chuan Shi and Yu, 2017) (llamada *"Heteregeneous Information Network"*, HIN, por sus siglas en inglés) o red heterogénea. Las redes heterogéneas describen sistemas complejos, éstas han incrementado los intereses investigativos de científicos de distintas áreas por su relevancia en sistemas reales como *world-wide web* (Deng, 2020), transportación (Chen et al., 2020), biología (Liu et al., 2020) y sistemas sociales (Balakrishnan and TV, 2020).

En las redes heterogéneas existe una subcategoría particular donde hay solo un tipo de objetos pero más de un tipo de relación entre objetos, esta se denomina red multirrelacional (M. K.-P. Ng and Ye). La red multirrelacional también es conocida como red multicapa donde cada capa representa un tipo de relación.

La determinación de la relevancia de los vértices o entidades que conforman la red es un crucial y retador tema que permite entender la estructura topológica y el proceso de cambio de los sistemas complejos de redes (Matos et al., 2018; Smolyak et al., 2020). Existen distintas medidas de centralidad basadas en un criterio ya sea estructural (tales como el grado y el coeficiente de agrupamiento), orientadas a las rutas que sigue la comunicación (como la Intermediación o Cercanía) y otros criterios más refinados (como el Vector Propio, PageRank y Katz). Las medidas de centralidad ayudan a una mejor identificación de los individuos que difunden y propagan la información en la red (Ribeiro et al., 2020), optimizar el uso de los recursos para facilitar la propagación de la información (Alshahrani et al., 2020), controlar el flujo de movilidad y reducir el impacto de la propagación de enfermedades (Jadidi et al., 2021), entre otras aplicaciones.

Las relaciones que se establecen entre entidades en una red suelen estar denotadas de distintas formas, lo que representa aristas de diferentes tipos, siendo este un sistema multirrelacional o multicapa. En los últimos años, las estrategias para manejar las redes multicapas combinan las múltiples interacciones entre los vértices considerando los tipos de aristas (Chuan Shi and Yu, 2017; De Bacco et al.; Kumar Behera et al.; Chen et al.; M. Wu and Poor). Varias medidas de centralidad en redes homogéneas han sido extendidas para identificar los nodos principales en redes multicapas tales como la medida de centralidad de grado, el PageRank y el Vector Propio.

En la última década, las redes sociales han crecido significativamente en cuanto a tamaño y actividad (dinamismo), provocando que los algoritmos estáticos para el cálculo de la centralidad sean muy costosos en

tiempo. Ante la dificultad que implica el tiempo de cálculo de las medidas de centralidad en redes de grandes dimensiones, este trabajo se propone como objetivo construir una herramienta para el análisis eficiente de estas medidas en una red multicapa.

Métodos o Metodología Computacional

Las principales funciones de una red social son: conectar a los usuarios y permitir la centralización tanto de información, como de recursos (fotos, videos, opiniones y otros), en un único lugar de fácil acceso e intercambio. Sin lugar a dudas las redes sociales han marcado una nueva pauta en la comunicación entre las personas, ya que éstas se han convertido en su principal medio de difusión, ya sea de sus actividades, como de su estado de ánimo. En este tipo de red existen múltiples tipos de conexiones entre las entidades que la conforman, lo cual la clasifica como una red multirrelacional o multicapa.

Una red multicapa se modela como una combinación de grafos que pertenecen a distintos niveles, los cuales se denominan capas, denotándose como $L_m = \{l_1, ..., l_m\}$ donde $G_\alpha^{L_m} = \{(V_\alpha, E_\alpha)^{L_m}\}$ representa el grafo en la capa α. Generalmente, el conjunto de usuarios es el mismo en las distintas capas; ejemplo: $V_\alpha = V_\beta = V, \forall \alpha, \beta \in L_m$. Por otra parte, una colección de aristas $E_\alpha \subseteq V_\alpha \times V_\alpha$ representan la interacción de una relación peculiar entre usuarios, tales como enlaces familiares, de amistad o colaboración (Chen et al.). Las relaciones pueden o no tener un nivel de relevancia o significación, lo cual se representa con un peso en la arista, este tipo de grafo se denomina grafo pesado o ponderado.

Medidas de Centralidad

Las medidas de centralidad denotan la importancia de las entidades, es decir, los vértices dentro de la red. Existen distintas medidas de centralidad que cuantifican la importancia de los nodos en cuanto a la cantidad de conexiones, con quiénes se conectan y papel que juegan en las comunicaciones. La identificación de individuos centrales en redes multicapas es una tarea investigada con intensidad, extendiendo el análisis clásico de las medidas de centralidad desde las redes de una sola capa.

La centralidad de ***Grado*** es la medida que posiciona a los vértices considerando el número de conexiones con otros vértices en la red. En los grafos dirigidos se definen dos medidas de centralidad de grado distintas, correspondientes al grado de entrada y al de salida. De forma análoga existen variantes de esta medida para grafos donde las aristas tienen un peso o importancia.

La medida de centralidad de ***Vector Propio*** es una medida de la influencia de un nodo en una red (Bonacich, 1987). Esta medida tiene un alto contraste con respecto a la centralidad de grado pues no todos los vecinos de un nodo son considerados como equivalentes. Los nodos tienen distintas relevancias y en esa medida contribuyen a la relevancia de otros nodos de la red. Una puntuación alta de Vector Propio significa que un nodo está conectado a muchos nodos que tienen puntuaciones altas. En grafos dirigidos puede calcularse utilizando la estrategia propuesta por (Newman, 2018), pero tiene como limitación que los nodos fuera de las componentes fuertemente conexas tengan centralidad igual a cero.

La medida de centralidad de ***Katz*** (Katz, 1953) es empleada para cuantificar el grado relativo de influencia de un actor en una red social. Tiene una idea conceptual muy parecida a la medida de Vector Propio, mas solventa la limitación de esta en redes orientadas. La medida garantiza que los nodos tengan un nivel de importancia distinto de cero sin importar si están dentro de una componente fuertemente conexa.

La medida de centralidad de ***PageRank*** es un modelo que tiene distintas interpretaciones (Gleich and Saunders, 2009). Se basa en la idea de que en una matriz estocástica, idealmente, la importancia de los nodos es proporcional al Vector Propio dominante que no es único. *PageRank* modifica este problema para producir un nuevo problema con una única respuesta. Como contraste a la centralidad *Katz*, el *PageRank* intenta diluir la importancia de un nodo con el resto de sus vecinos.

La medida de centralidad ***Cercanía***, describe cuán cerca se encuentra un vértice de los otros nodos en la red. En redes donde existan elementos en distintas componentes conexas, se tiene que la distancia entre dichos vértices es infinita. Existen otras alternativas para remplazar la distancia infinita entre dos vértices de componentes distintas, como lo es centralidad armónica (Rochat, 2009).

La ***Intermediación*** al igual que la *Cercanía* define la importancia de un vértice evaluando su papel en las comunicaciones, cuantificando las mejores rutas en las que aparece un nodo. Esta medida es tiene un alto costo computacional debido al cálculo que implica calcular todos los caminos mínimos, el algoritmo de Brandes (Brandes, 2001) es la estrategia óptima reportada para su cálculo.

Existen distintos enfoques para calcular las medidas de centralidad en redes multicapas, que se pueden clasificar en 3 categorías principales (Chen et al.):

- Los que suman la información de las distintas capas y luego estiman la medida de centralidad en tal red (F. Battiston and Latora; M. De Domenico and Arenas).

- Aquellos que tratan de forma independiente las capas y evalúan la centralidad en cada capa (De Domenico et al., 2015).
- Los que consideran la interacción entre capas (Reiffers-Masson and Labatut; M. Wu and Poor).

Herramientas

El lenguaje Scala se desarrolló en el 2001 en la universidad EPFL (Escuela Politécnica Federal de Lausana en Suiza). Scala ha experimentado un acentuado crecimiento que ha hecho que este lenguaje pase de ser utilizado de un modo muy académico y orientado a la investigación, a convertirse en un estándar para muchas empresas (algunas de la envergadura de Twitter o BBVA, empresas en creación y universidades de todo el mundo.

Apache Spark es un *framework* de código abierto para almacenar y procesar grandes cantidades de información distribuida en un clúster de ordenadores. Spark puede realizar todas las operaciones en memoria, pudiendo llegar a operar hasta 100 veces más rápido que Apache Hadoop (Xu et al., 2017). Además de las operaciones *Map* y *Reduce*, Spark incluye una nueva estructura de datos: *Resilient Distributed Datasets* o RDD sobre la cual Apache Spark realiza todas las operaciones de forma paralela y distribuida.

Spark utiliza otra abstracción computacional, las variables compartidas, que pueden ser utilizadas en operaciones paralelas. Las variables compartidas es el mecanismo que se emplea para enviar los datos necesarios a los nodos ejecutores o entre ejecutor y driver. En Spark existen dos tipos de variables compartidas: las *broadcast*, las cuales pueden ser utilizadas como caché en la memoria de todos los nodos, y los *acumulator*, los cuales son variables que solo se les puede incrementar como contadores o sumadores.

Las transformaciones son computadas de modo perezoso (*lazy*), es decir, solo cuando una acción es invocada, la cual desata el cálculo de las transformaciones luego de que Spark cree el plan físico de ejecución. En Spark los datos no son generalmente distribuidos en particiones de modo que estén en el lugar justo para una operación específica. El *shuffling* consiste en localizar todos los valores para una clave que no residen en la misma partición o en la misma máquina.

Diseño de la propuesta

En el cálculo de la medida de centralidad se emplea una combinación de los enfoques: (1) tratar la centralidad por capa independiente y (2) analizar la suma de información de las capas. Dada la medida de centralidad

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Mes Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-17

ϕ, el vértice x_i y una selección L_s en un grafo multicapa $G^{L_m} = \{(V,E)_{l_1}, ..., (V,E)_{l_m}\} = \{(V,E)^{L_m}\}$ donde $L_m = \{l_1, ..., l_m\}$ y $L_s \subseteq L_m$, denotaremos la medida de centralidad del vértice en la selección como:

$$\phi^{L_s}(x_i, G^{L_s}) = \begin{pmatrix} \phi(x_i, G^{L_s}_{l_1}) \\ \vdots \\ \phi(x_i, G^{L_s}_{l_s}) \\ \phi(x_i, G^{L_s}_{l_{merge}} = \{\bigcup_{i=1}^{s}(V,E)_{l_i}\}) \end{pmatrix} = \begin{pmatrix} \varphi_{x_i}^{l_1} \\ \vdots \\ \varphi_{x_i}^{l_s} \\ \varphi_{x_i}^{l_{merge}} \end{pmatrix} \quad (1)$$

en donde $\phi(x_i, G^{L_s}_{l_j}) = \varphi_{x_i}^{l_j}$ representa la centralidad ϕ para el vértice x_i en el grafo homogéneo formado por la capa $l_j \in L_s$, mientras que $\phi(x_i, G^{L_s}_{l_{merge}}) = \varphi_{x_i}^{l_{merge}}$ lo es para el grafo homogéneo de la unión de las capas L_s.

El principal propósito de emplear Scala usando el *framework* Spark es aprovechar las ventajas que nos brinda para el cómputo paralelo y distribuido. Las distintas medidas que son expuestas en este documento se calculan de forma paralela y distribuida, con el empleo de la estructura de grafo propuesta en *GraphX*. La estrategia consiste en distribuir el cómputo de las medidas en distintas unidades de cómputo, donde cada ejecutor efectúa el cálculo de forma seriada por cada nodo, tal como se muestran en el algoritmo 1.

Algoritmo 1 Cálculo paralelo y distribuido de una métrica

Procedimiento COMPUTODISTRIBUIDOPARALELO($G^{L_m} = \{(V,E)_{l_1}, ..., (V,E)_{l_m}\}, L_s, \phi$)
  $\varphi[] \leftarrow$ Inicializar cada par (vértice, capa) con valor mínimo (0)
  Selección $\leftarrow$ Filtrar las capas de G^{L_m} considerando L_s
  Compartir para cada unidad de cómputo *Selección*
  Para Cada Unidad de cómputo *Distribuida* **Hacer**
    Para Cada vértice $(x_i, l_s) \in$ *Selección*, de forma *Paralela* **Hacer**
      $\varphi[x_i, l_s] \leftarrow$ calcular $\phi(x_i, G^{L_s}_{l_s})$
    Fin Para
  Fin Para
  Retornar φ
Fin Procedimiento

Evaluación

Para la evaluación de las medidas de centralidad implementadas se emplearon colecciones de grafos multicapas de distintas naturaleza (tabla 1). La red ARABIDOPSIS está conformada por diferentes tipos de in-

teracciones genéticas (BioGRID[1]) (Stark et al., 2006). Las colecciones NYCLIMATE y CANNES fueron obtenidas de la red social Twitter considerando diferentes tipos de relaciones sociales entre usuarios (Omodei et al., 2015).

Nombre de la Colección	Naturaleza de la Red	Número de Nodos	Número de Aristas	Número de Capas	Número de Nodos a Analizar	Número de Aristas a Analizar	Número de Capas a Analizar
ARABIDOPSIS	Biológica	6 980	18 655	7	55 840	37 310	8 (7 + Capa Mezcla)
NYCLIMATE	Social	102 439	353 495	3	409 756	706 990	4 (3 + Capa Mezcla)
CANNES	Social	438 537	991 854	3	1 754 148	1 983 708	4 (3 + Capa Mezcla)

Tabla 1 - Propiedades de los grafos multicapas y su transformación para la experimentación.

Al evaluar algoritmos paralelos y distribuidos hay que considerar la aceleración y escalabilidad (Gil, 2005):

- T_{dist} : La duración del algoritmo paralelo y distribuido propuesto, considerada como el tiempo transcurrido desde que se inicia su ejecución hasta que finaliza el último de sus procesos.
- T_{sec} : La duración del algoritmo secuencial para resolver el problema tratado. En nuestro contexto, se utilizó la implementación propia en el lenguaje *Scala* con un único núcleo.
- Aceleración: $A = \frac{T_{sec}}{T_{dist}}$, mide el incremento de velocidad obtenido con el algoritmo paralelo y distribuido. En el caso ideal crece linealmente con el número de los recursos.
- Escalabilidad vertical: Cuando al añadir más recursos a los nodos del sistema hay un incremento de velocidad en el algoritmo.
- Escalabilidad horizontal: Cuando al agregar más nodos al sistema el rendimiento del algoritmo mejora.

Para la ejecución de los distintos experimentos realizados se utilizó el Supercomputador de la Universidad de Oriente, en particular el clúster Big Data. Este clúster, especializado en el cómputo de grades volúmenes de datos, virtualmente se compone de 3 unidades de cómputo formadas por 64 núcleos y 64GB de memoria RAM donde cada una tiene 600GB de almacenamiento. El clúster Big Data cuenta con la versión 2.11.8 del lenguaje Scala y el framework Spark en su versión 2.3.2.

[1] www.thebiogrid.org

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Mes Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-17

Resultados y discusión

Con el propósito de analizar el comportamiento de los algoritmos implementados se emplearon las distintas colecciones anteriormente mencionadas y se ejecutaron un conjunto de pruebas. De igual manera se evaluó el rendimiento de las diferentes medidas sin considerar el peso de las aristas.

Cada uno de los experimentos realizados se ejecutó en el ambiente del Supercomputador de la Universidad de Oriente. En los experimentos efectuados con cada una de las diferentes métricas se emplearon configuraciones de 1, 2, 4, 8 y 16 núcleos utilizando 3GB de memoria RAM en un único esclavo, con el propósito de evaluar la escalabilidad vertical. En las configuraciones de 32, 64 y 128 se emplearon 2, 4 y 8 esclavos respectivamente empleando 3GB de memoria RAM en cada esclavo, con el objetivo de evaluar la escalabilidad horizontal de la solución propuesta.

En las distintas colecciones se realizaron experimentos donde se analizaron todas las capas además de su mezcla. Lo anteriormente mencionado implica calcular para cada vértice su relevancia en todas las capas, este cálculo se realiza de forma paralela y distribuida empleando los algoritmos de menor complejidad computacional reportados en la literatura.

Experimentos con la métrica Cercanía

Los experimentos realizados con la medida de centralidad de Cercanía en las distintas colecciones descritas en la tabla 1 se pueden ver en las siguientes tablas. En la tabla 2, así como en la figura 1a se muestra el comportamiento de los tiempos de ejecución de la métrica de Cercanía sin el peso de las arista.

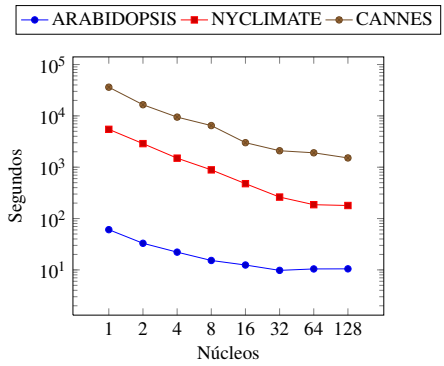
Nombre de la Colección	1 Núcleo 1 Ejecutor	2 1 Ejecutor	4 Núcleos 1 Ejecutor	8 Núcleos 1 Ejecutor	16 Núcleos 1 Ejecutor	32 Núcleos 2 Ejecutores	64 Núcleos 4 Ejecutores	128 Núcleos 8 Ejecutores
ARABIDOPSIS	35.237	20.497	14.161	10.288	11.073	8.328	7.366	6.542
NYCLIMATE	3519.209	1604.342	871.457	536.832	292.003	254.561	136.548	122.829
CANNES	36178.061	16479.062	9459.235	6474.781	3006.591	2094.340	1908.696	1515.883

Tabla 2 - Tiempos en segundos para la ejecución de la métrica de Cercanía sin el peso de las aristas.

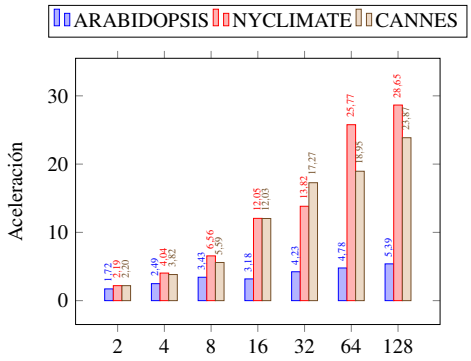
La tabla 2 refleja el comportamiento del tiempo para la medida de Cercanía sin el peso de las aristas. Se puede apreciar que al aumentar las capacidad de cómputo se reduce el tiempo de ejecución, ver figura 1a. Los tiempos en un ejecutor con 1, 2, 4, 8 y 16 núcleos muestran que al aumentar los recursos del ejecutor se reduce el tiempo, lo que evidencia que escala verticalmente. De forma similar sucede que al aumentar la cantidad de ejecutores se reduce el tiempo de cómputo, por lo que podemos afirmar que escala horizontalmente.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Mes Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-17

En la figura 1b se grafica el comportamiento de la aceleración para la métrica de Cercanía en las tres colecciones para la variante sin peso. En las figura la colección ARABIDOPSIS es la de peor desempeño, alcanzando una aceleración de 5.38 veces con 128 núcleos para la variante sin analizar el peso de las aristas, esto se debe al intercambio de información entre ejecutores y el *driver*. Por otra parte, en la figura 1b se obtiene una aceleración de 28.65 y 23.86 veces en las colecciones de NYCLIMATE y CANNES respectivamente ambas con 128 núcleos distribuidos en 8 esclavos.



(a) Tiempos de ejecución en segundos, métrica Cercanía no ponderada



(b) Aceleración de la medida Cercanía no ponderada

Fig. 1 - Tiempo de ejecución y aceleración para la métrica de Cercanía no ponderada

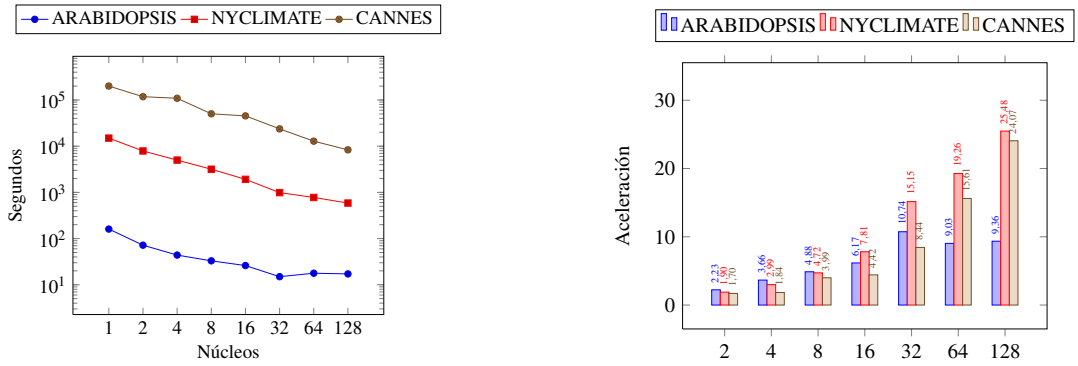
Experimentos con la métrica Intermediación

La tabla 3 y la figura 2a refleja el comportamiento del tiempo en la medida de Intermediación sin el peso de las aristas. Al aumentar la capacidad de cómputo se reduce el tiempo de ejecución para la mayoría de las colecciones, excepto en ARABIDOPSIS que con 64 núcleos en 4 ejecutores tiene un desempeño menor que con 32 en 2 ejecutores. En la colección ARABIDOPSIS no se logra acelerar más luego de 32 núcleos y esto ocurre debido al costo que tiene la intercomunicación entre los ejecutores. De forma general, los tiempos en un ejecutor con 1, 2, 4, 8 y 16 núcleos demuestran que al aumentar los recursos del ejecutor se reduce el tiempo, lo que evidencia que escala verticalmente. De forma similar sucede que al aumentar la cantidad de ejecutores se reduce el tiempo de cómputo, por lo que podemos afirmar que escala horizontalmente.

Nombre de la Colección	1 Núcleo 1 Ejecutor	2 Núcleos 1 Ejecutor	4 Núcleos 1 Ejecutor	8 Núcleos 1 Ejecutor	16 Núcleos 1 Ejecutor	32 Núcleos 2 Ejecutores	64 Núcleos 4 Ejecutores	128 Núcleos 8 Ejecutores
ARABIDOPSIS	160.630	71.948	43.923	32.940	26.040	14.955	17.783	17.170
NYCLIMATE	14968.183	7886.632	5002.114	3174.521	1917.524	987.726	777.116	587.399
CANNES	200657.657	118017.119	109113.904	50254.848	45350.050	23765.894	12851.306	8338.010

Tabla 3 - Tiempos en segundos para la ejecución de la métrica de Intermediación no ponderada.

En la figura 2b se muestra el comportamiento de la aceleración para la métrica de Intermediación en las tres colecciones para la variante sin peso. La colección ARABIDOPSIS es la de menor desempeño, consiguiendo una aceleración de 9.35 veces con 128 núcleos. Se alcanzan los valores de aceleración de 24.06 y 25.48 veces en las colecciones de NYCLIMATE y CANNES respectivamente, ambas con 128 núcleos distribuidos en 8 esclavos con 16 núcleos cada uno.



(a) Tiempos de ejecución en segundos, métrica Intermediación no ponderada.

(b) Aceleración de la medida Intermediación no ponderada

Fig. 2 - Tiempo de ejecución y aceleración para la métrica de Intermediación no ponderada

Experimentos con la métrica Vector Propio

En la tabla 4 y figura 3a está reflejado el comportamiento del tiempo en la medida de Vector Propio sin el peso de las aristas, en donde se puede apreciar que al aumentar la capacidad de cómputo se reduce el tiempo de ejecución. Los tiempos en un ejecutor y las distintas configuraciones de núcleos ilustran que al aumentar los recursos del ejecutor se reduce el tiempo, lo que evidencia que escala verticalmente. De forma similar sucede que al aumentar el número de ejecutores se reduce el tiempo de cómputo, por lo que podemos afirmar que escala horizontalmente.

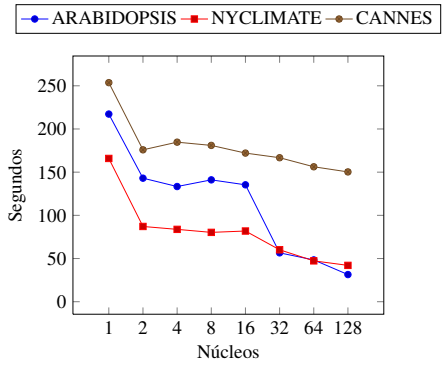
Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Mes Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-17

Colección	Núcleo(s)	1	2	4	8	16	32	64	128
	Ejecutor(es)	1	1	1	1	1	2	4	8
ARABIDOPSIS	217.26		143.03	133.35	141.08	135.34	56.57	48.38	31.43
NYCLIMATE	165.88		87.12	83.76	80.28	81.84	60.12	47.30	42.06
CANNES	253.64		175.85	184.72	180.98	172.08	166.723	156.234	150.284

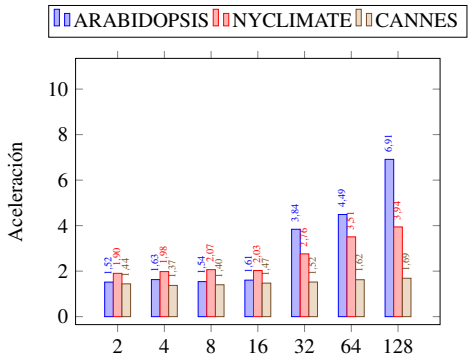
Tabla 4 - Tiempos en segundos para la ejecución de la métrica de Vector Propio sin considerar el peso de las aristas.

En la figura 3b se grafica el comportamiento de la aceleración para la métrica de Vector Propio en las tres colecciones. La colección ARABIDOPSIS es la de mejor desempeño a diferencia de lo que había ocurrido con las métricas anteriores. Este desempeño se debe a que solo el 29 % de sus nodos tiene relevancia nula mientras que para las otras dos colecciones este valor es del 89 % aproximadamente, por lo que en ARABIDOPSIS existe una mayor cantidad de elementos a computar de forma paralela y distribuida. En ARABIDOPSIS se alcanza una aceleración de 6.91 con la configuración de 128 núcleos entre 8 ejecutores.

El desempeño de la métrica en la colección NYCLIMATE consiguió una aceleración para la variante sin considerar el peso de las aristas de 3.94 veces con 128 núcleos en 8 ejecutores. La colección CANNES alcanza una aceleración de 1.69 veces con el empleo de 128 núcleos en 8 ejecutores, tal como se ilustra en la figura 3b. Aunque esta medida escala horizontal y verticalmente su naturaleza secuencial dificulta que se alcancen mejores rendimientos en su implementación paralela y distribuida.



(a) Tiempos de ejecución en segundos, métrica Vector Propio no ponderada.



(b) Aceleración de la medida Vector Propio no ponderada

Fig. 3 - Tiempo de ejecución y aceleración para la métrica de Vector Propio no ponderada

Conclusiones

En este trabajo se desarrolló una herramienta para la evaluación de las medidas de centralidad en redes multicapas. La principal utilidad de la propuesta es obtener niveles de relevancia en correspondencia del nivel semántico definido por los tipos de aristas y su combinación en un tiempo óptimo. Para la optimización del tiempo de cómputo se empleó el lenguaje Scala con el *framework* Spark para el cómputo paralelo y distribuido de las medidas, empleando los algoritmos de menor complejidad computacional reportados en la literatura.

Como parte de los experimentos desarrollados se evaluó el desempeño de estas medidas en el ambiente del Supercomputador de la Universidad de Oriente. En los experimentos efectuados con las diferentes métricas se emplearon configuraciones de 1, 2, 4, 8 y 16 núcleos utilizando 3GB de memoria RAM en un único esclavo, los cuales demostraron que las medidas de centralidad implementadas escalan verticalmente. Además, en las configuraciones de 32, 64 y 128 se emplearon 2, 4 y 8 esclavos respectivamente con 3GB de memoria RAM en cada uno, demostrando también que las medidas logran escalar horizontalmente.

En los experimentos realizados se aceleró el tiempo de cómputo para la métrica de Cercanía unas 28.65 veces, empleando 128 núcleos en 8 esclavos con 3GB de memoria RAM cada uno. De similar forma, la medida de Intermediación acelera 25.48 veces con igual configuración de recursos. Por otra parte y en menor medida se aceleró el cómputo de la métrica de Vector Propio con una aceleración de 6.91 veces, esta menor aceleración se deben a la naturaleza secuencial de su cómputo y al fenómeno del *shuffling*.

Referencias

Mohammed Alshahrani, Zhu Fuxi, Ahmed Sameh, Soufiana Mekouar, and Sheng Huang. Efficient algorithms based on centrality measures for identification of top-k influential users in social networks. *Information Sciences*, 527:88–107, 2020.

M Balakrishnan and G TV. A neural network framework for predicting dynamic variations in heterogeneous social networks. *PLoS ONE*, 15(4):e0231842, 2020.

Phillip Bonacich. Power and centrality: A family of measures. *American journal of sociology*, 92(5):1170–1182, 1987.

Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2): 163–177, 2001.

Jianjun Chen, Yue Deng, Zhen Su, Songxin Wang, Chao Gao, and Xianghua Li. Identifying multiple influential users based on the overlapping influence in multiplex networks. *IEEE Access*, 7:156150–156159. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2949678.

Zhibin Chen, Xi Lin, Yafeng Yin, and Meng Li. Path controlling of automated vehicles for system optimum on transportation networks with heterogeneous traffic stream. *Transportation Research Part C: Emerging Technologies*, 110:312–329, 2020.

Jiawei Zhang Yizhou Sun Chuan Shi, Yitong Li and Philip S. Yu. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, pages 17–37, 2017.

Caterina De Bacco, Eleanor A. Power, Daniel B. Larremore, and Cristopher Moore. Community detection, link prediction, and layer interdependence in multilayer networks. *Physical Review E*, 95(4):042317. ISSN 2470-0045, 2470-0053. doi: 10.1103/PhysRevE.95.042317. URL http://arxiv.org/abs/1701.01369.

Manlio De Domenico, Albert Solé-Ribalta, Elisa Omodei, Sergio Gómez, and Alex Arenas. Ranking in interconnected multilayer networks reveals versatile nodes. *Nature communications*, 6(1):1–6, 2015.

Weiwei Deng. Leveraging consumer behaviors for product recommendation: an approach based on heterogeneous network. *Electronic Commerce Research*, pages 1–27, 2020.

V. Nicosia F. Battiston and V. Latora. Structural measures for multiplex networks. *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip*, 89(3).

Reynaldo Gil. *Algoritmos de Agrupamiento sobre Grafos y su Paralelización*. PhD thesis, Tesis doctoral en Ciencia de la Computación, Universidad Jaume I, España, 2005.

David Francis Gleich and Michael Saunders. *Models and algorithms for pagerank sensitivity*. Stanford University Stanford, CA, 2009.

MohammadMohsen Jadidi, Saeed Jamshidiha, Iman Masroori, Pegah Moslemi, Abbas Mohammadi, and Vahid Pourahmadi. A two-step vaccination technique to limit covid-19 spread using mobile data. *Sustainable Cities and Society*, 70:102886, 2021.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

Ranjan Kumar Behera, Santanu Kumar Rath, Sanjay Misra, Robertas Damaševičius, and Rytis Maskeliūnas. Distributed centrality analysis of social network data using MapReduce. *Algorithms*, 12(8):161. ISSN 1999-4893. doi: 10.3390/a12080161. URL https://www.mdpi.com/1999-4893/12/8/161.

Chuang Liu, Yifang Ma, Jing Zhao, Ruth Nussinov, Yi-Cheng Zhang, Feixiong Cheng, and Zi-Ke Zhang. Computational network biology: data, models, and applications. *Physics Reports*, 846:1–66, 2020.

S. Gómez M. De Domenico, A. Solé-Ribalta and A. Arenas. Navigability of interconnected networks under random failures. *Proc. Nat. Acad. Sci. USA*, 111(23):8351–8356.

X. Li M. K.-P. Ng and Y. Ye. Multirank: Co-ranking for objects and relations in multi-relational data. *17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 8(10):1217–1225.

Y. Zhang J. Chen Y. Sun Y.-Y. Liu J. Zhang M. Wu, S. He and H. V. Poor. A tensor-based framework for studying eigenvector multicentrality in multilayer networks. *Proc. Nat. Acad. Sci. USA*, 116(31):15407–15413.

Armando Díaz Matos, Reynaldo Gil Pons, and Reynier Ortega Bueno. Predicción de la evolución de comunidades en redes sociales. *Revista Cubana de Ciencias Informáticas*, 12(4):115–127, 2018.

Mark Newman. *Networks*. Oxford university press, 2018.

Elisa Omodei, Manlino De De moneico, and Alex Arenas. Characterizing interactions in online social networks during exceptional events. *Frontiers in Physics*, page 59, 2015.

A. Reiffers-Masson and V. Labatut. Opinion-based centrality in multiplex networks: A convex optimization approach. *Netw. Sci*, 5(2):213–234.

Ana Carolina Ribeiro, Bruno Azevedo, Jorge Oliveira e Sá, and Ana Alice Baptista. How to measure influence in social networks? In *International Conference on Research Challenges in Information Science*, pages 38–57. Springer, 2020.

Yannick Rochat. Closeness centrality extended to unconnected graphs: The harmonic centrality index. Technical report, 2009.

Alex Smolyak, Orr Levy, Irena Vodenska, Sergey Buldyrev, and Shlomo Havlin. Mitigation of cascading failures in complex networks. *Scientific reports*, 10(1):1–12, 2020.

Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, (suppl_1):D535–D539, 2006.

Bo Xu, Changlong Li, Hang Zhuang, Jiali Wang, Qingfeng Wang, and Xuehai Zhou. Efficient distributed smith-waterman algorithm based on apache spark. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 608–615. IEEE, 2017.

Contribuciones de los autores

1. Conceptualización: Armando Díaz Matos
2. Curación de datos: Armando Díaz Matos
3. Análisis formal: Armando Díaz Matos y Darian Horacio Grass Boada
4. Adquisición de fondos: DATYS y Armando Díaz Matos
5. Investigación: Armando Díaz Matos
6. Metodología: Armando Díaz Matos
7. Administración del proyecto: Armando Díaz Matos y Darian Horacio Grass Boada
8. Recursos: Armando Díaz Matos
9. Software: Armando Díaz Matos
10. Supervisión: Darian Horacio Grass Boada
11. Validación: Armando Díaz Matos
12. Visualización: Armando Díaz Matos
13. Redacción - borrador original: Armando Díaz Matos
14. Redacción - revisión y edición: Darian Horacio Grass Boada y Armando Díaz Matos