Convención Científica Internacional UCLV 2021

Transferencia de Conocimiento en Tecnologías de la Información

Herramienta para el cálculo de la centralidad en redes multicapas
(A new framework for centrality on multilayer networks)

Armando Díaz Matos [a], Darian Horacio Grass Boada [b]

[a] *Centro de Desarrollo de Aplicaciones, Tecnologías y Sistemas (DATYS). s/n, Avenida Patricio Lumumba, Santiago de Cuba, Cuba.*

[b] *Centro de Aplicaciones de Tecnologías de Avanzada. 7ma A #21406 E/ 214 y 216, Rpto. Siboney, Playa, La Habana, Cuba.*

Resumen

En este trabajo se presenta una herramienta para el cálculo de las medidas de centralidad de cercanía e intermediación 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 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. Los experimentos realizados alcanzaron niveles de aceleración de hasta 30.47 y 25.48 veces para la medida de Cercanía e Intermediación respectivamente. Los resultados demuestran que las medidas de centralidad implementadas escalan tanto vertical como horizontalmente.

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

1. 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 [5]. Cuando la información que se maneja por las entidades o sus relaciones presentan diferencias en su tipo se puede afirmar que es una red

Email address: armando.diaz@datys.cu (Armando Díaz Matos).

.

de información heterogénea [5] (llamada *"Heteregeneous Information Network"*, HIN, por sus siglas en inglés) o red heterogénea.

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 [13]. La red multirrelacional también es conocida como red multicapa donde cada capa representa un tipo de relación. Este trabajo se enfoca en el análisis de este tipo de red.

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 [15, 20, 3]. Existen distintas medidas de centralidad que cuantifican la influencia de los vértices en la red en general. Las medidas de centralidad pueden ayudar a una mejor identificación de los individuos que difunden y propagan la información dentro de la red [18], controlar el flujo de movilidad y reducir el impacto de la propagación de enfermedades [10], 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 [1, 5, 6, 11, 4, 14].

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. Los algoritmos de centralidad de cercanía e intermediación son los de mayor complejidad computacional por lo que su cómputo resulta más costoso. Ante la dificultad que implica el tiempo de cálculo de las medidas de centralidad de cercanía e intermediación 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.

2. Metodología

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 [4]. 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.

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, de ahí que la medida de Cercanía tiende a 0. Existen otras alternativas para remplazar la distancia infinita entre dos vértices de componentes distintas, una de ellas es la centralidad armónica [19].

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. El algoritmo de Brandes [2] calcula esta medida de forma óptima.

Existen distintos enfoques para calcular las medidas de centralidad en redes multicapas, que se pueden clasificar en 3 categorías principales [4]:

- Los que suman la información de las distintas capas y luego estiman la medida de centralidad en tal red [8, 12].
- Aquellos que tratan de forma independiente las capas y evalúan la centralidad en cada capa [7].
- Los que consideran la interacción entre capas [17, 14].

El lenguaje Scala se desarrolló en el 2001 en la universidad EPFL [1] (Suiza) y se hizo público desde el mundo académico por parte de Martin Odersky en el año 2003. 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), *startups* [2] 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 "corrientes", entendiendo por corrientes los ordenadores que se pueden tener en casa, y no superordenadores. Spark soluciona los problemas de su contrincante Apache Hadoop haciendo todas las operaciones en memoria y pudiendo llegar a operar hasta 100 veces más rápido que éste, [22]. Además de las operaciones *Map* y *Reduce* que Apache Hadoop ofrece, Apache 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.

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 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:

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

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

[1] Escuela Politécnica Federal de Lausana.
[2] Empresas que se encuentran en edad temprana o nueva creación con tendencias de crecimiento.

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

procedure 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 procedure

En los algoritmos paralelos y distribuidos es necesario que con el aumento de la cantidad de recursos los tiempos de cómputo disminuyan, o al menos es necesario que se puedan procesar mayores volúmenes de datos. Estas características deseables de los algoritmos paralelos y distribuidos se llaman aceleración y escalabilidad [9]. Con el objetivo de evaluar estas prestaciones en los algoritmos propuestos se consideran, en primer lugar, dos valores básicos:

- T_{dist}: La duración del algoritmo paralelo y distribuido propuesto.
- T_{sec}: La duración del algoritmo secuencial para resolver el problema tratado. En nuestro contexto, se utilizó la implementación propia con un único núcleo.

Algunas de las medidas más utilizadas para evaluar la eficiencia y escalabilidad de un algoritmo paralelo y distribuido son:

- 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 evaluación de las medidas de centralidad implementadas se emplearon colecciones de grafos multicapas de distintas naturaleza. La red ARABIDOPSIS está conformada por diferentes tipos de interacciones genéticas (BioGRID [3]) [21]. Las colecciones NYCLIMATE y CANNES fueron obtenidas de la red social Twitter considerando diferentes tipos de relaciones sociales entre usuarios [16]. En la tabla 1 se muestran las principales características de las distintas colecciones empleadas para los experimentos,

[3] `www.thebiogrid.org`

además de las propiedades que toman luego de aplicar la transformación propuesta por nuestro modelo de análisis para redes multicapas.

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

Cuadro 1
Propiedades de las redes y su transformación para la experimentación.

Para la ejecución de los experimentos se empleó el Supercomputador de la Universidad de Oriente, en particular el clúster Big Data. Este clúster, especializado en el cómputo de grades volúmenes de datos cuyas especificaciones físicas pueden verse en la tabla 2, 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 de Scala y Spark en su versión 2.3.2.

Servidor	Cantidad	Procesador	Núcleos	RAM/núcleo	Red
SUN-FIRE x4470 M2	2	Intel Xeon X7550	64	64GB	1Gbps
DELL R720	1	Intel Xeon E5-2609	64	48GB	1Gbps
SUN-FIRE x4140	2	AMD Optetron 2431	12	48GB	1Gbps

Cuadro 2
Propiedades físicas del clúster Big Data de la Universidad de Oriente.

3. Resultados y discusión

En cada colección se ejecutaron un conjunto de pruebas con el objetivo de evaluar el comportamiento de estos algoritmos. De igual manera se evaluó el rendimiento de las diferentes medidas considerando o no el peso de las aristas. Los distintos resultados que se muestran a continuación tienen como propósito ilustrar el desempeño de los distintos algoritmos ante la variación de la capacidad de cómputo.

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 las tablas 3 y 4, así como en la figuras 1a y 1b se muestra el comportamiento de los tiempos de ejecución de la métrica de Cercanía sin el peso de las aristas y considerando el peso de las aristas respectivamente.

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	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

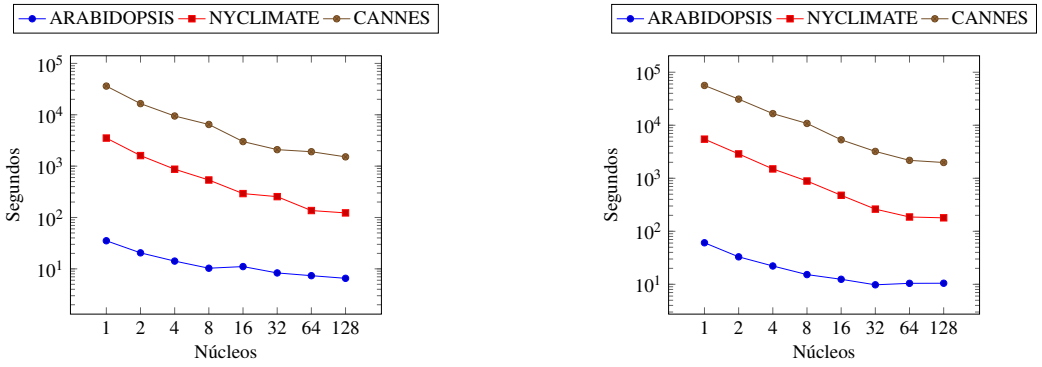
Cuadro 3
Tiempos en segundos para la ejecución de la métrica de Cercanía sin el peso de las aristas.

La tabla 3 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.

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	60.726	32.986	22.168	15.257	12.443	9.787	10.436	10.480
NYCLIMATE	5468.888	2879.716	1500.797	887.486	477.797	261.965	186.171	179.461
CANNES	56305.827	31092.857	16584.749	10820.544	5318.912	3210.841	2177.268	1983.825

Cuadro 4
Tiempos en segundos para la ejecución de la métrica de Cercanía considerando el peso de las aristas.

De similar forma se comportan los resultados en la tabla 4, donde se ilustra el tiempo de la medida considerando el peso de las aristas. 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, pues al aumentar la cantidad de cores se distribuyen los ejecutores distintos ordenadores. Es esta colección la de menor tamaño por lo que su cómputo se efectúa más rápido pero al estar los ejecutores distribuidos en distintos ordenadores, el intercambio de información entre ejecutores y el *driver* atenta en mayor medida contra el desempeño. Los experimentos evidencian que la medida escala tanto vertical como horizontalmente, ver figura 1b.

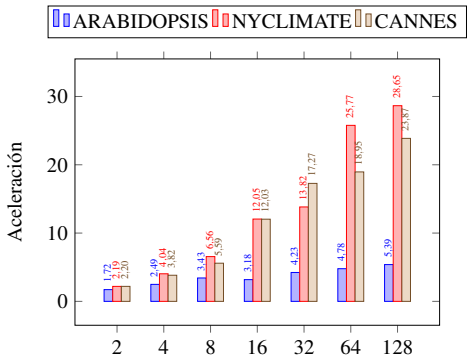


(a) Tiempos de ejecución para la métrica Cercanía sin peso

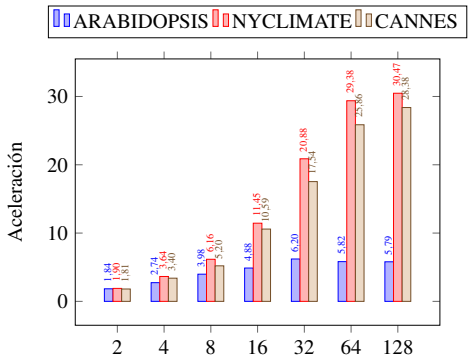
(b) Tiempos de ejecución para la métrica Cercanía con peso

Figura 1. Tiempos de ejecución en segundos por colecciones para la métrica Cercanía.

En la figura 2 se grafica el comportamiento de la aceleración para la métrica de Cercanía en las tres colecciones, tanto para la variante sin peso (figura 2a) como la pesada (figura 2b). En las figuras 2a y 2b 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. Análogamente, la variante ponderada tiene una aceleración de 6.20 veces con el empleo de 32 núcleos distribuidos en 2 esclavos con 16 núcleos cada uno, esto se debe al intercambio de información entre ejecutores y el *driver*.



(a) Aceleración para Cercanía sin el peso de las aristas



(b) Aceleración para Cercanía considerando el peso de las aristas

Figura 2. Aceleración de la medida Cercanía

Por otra parte, en la figura 2a 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. De forma similar, la versión que considera el peso de las arista, tal como se muestra en la figura 2b, alcanza una aceleración de 30.47 y 28.38 veces respectivamente con igual configuración.

La tabla 5 y la figura 3a 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.

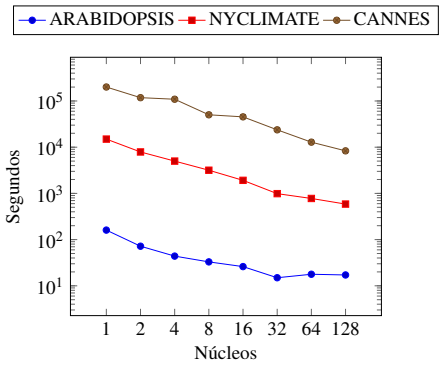
Nombre de la Colección	**1 Núcleo 1 Ejecutor**	**2 Núcleos 1 Ejecutor**	**4 Núcleos 1 Ejecutor**	**8 Núcleos 1 Ejecutor**	**16 Núcleos 1 Ejecutor**	**32 Núcleos 2 Ejecutores**	**64 Núcleos 4 Ejecutores**	**128 Núcleos 8 Ejecutores**
ARABIDOPSIS	160.630	71.948	43.923	32.940	26.040	14.955	17.783	17.170
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

Cuadro 5
Tiempos en segundos para la ejecución de la métrica de Intermediación sin el peso de las aristas.

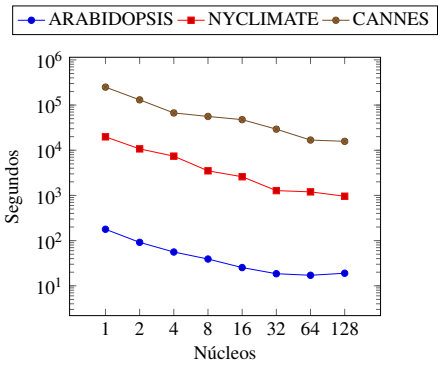
Con similar comportamiento están los resultados en la tabla 6, donde se analiza el tiempo de la medida considerando el peso de las aristas. Al aumentar las capacidad de cómputo se reduce el tiempo de ejecución para la mayoría de las colecciones, tal como se ilustra en la figura la figura 3b. De igual forma se evidencia que la implementación de la métrica escala tanto horizontal como verticalmente en cada una de las colecciones.

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	178.448	91.610	55.967	39.145	25.326	18.525	17.104	18.980
NYCLIMATE	19811.715	10726.360	7399.074	3506.097	2591.699	1278.765	1199.555	960.252
CANNES	249463.277	130449.774	66952.870	56009.568	47556.976	29253.135	16819.666	15749.418

Cuadro 6
Tiempos en segundos para la ejecución de la métrica de Intermediación considerando el peso de las aristas.



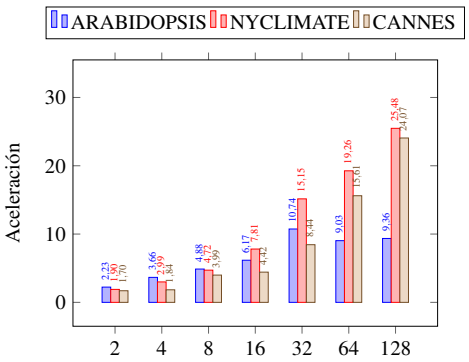
(a) Tiempos de ejecución para la métrica Intermediación sin peso



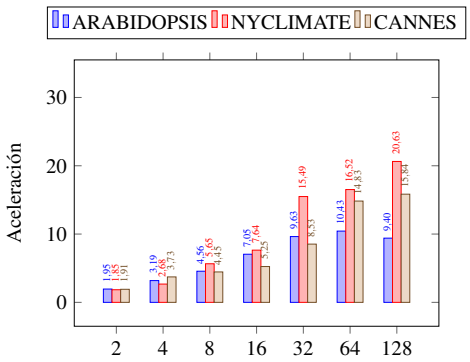
(b) Tiempos de ejecución para la métrica Intermediación con peso

Figura 3. Tiempos de ejecución en segundos por colecciones para la métrica Intermediación.

En la figura 4 se muestra el comportamiento de la aceleración para la métrica de Intermediación en las tres colecciones, tanto para la variante sin peso (figura 4a) como la pesada (figura 4b). En la figuras 4a y 4b, la colección ARABIDOPSIS es la de menor desempeño, consiguiendo una aceleración de 9.35 veces con 128 núcleos sin considerar el peso de las aristas. En esta misma colección la variante ponderada acelera 10.43 veces con 64 núcleos en 4 ejecutores, al aumentar la cantidad de ejecutores y nodos no se obtienen menores tiempos, esto se debe a la sobrecarga que implica la comunicación entre los distintos ejecutores y el máster, similar al comportamiento mostrado en la métrica de cercanía ponderada para esta misma colección.



(a) Aceleración para Intermediación sin el peso de las aristas



(b) Aceleración para Intermediación considerando el peso de las aristas

Figura 4. Aceleración de la medida Intermediación

En la figura 4a 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. Mientras que en la versión donde se considera el peso de las aristas la aceleración es 20.63 y 16.33 veces para NYCLIMATE y CANNES respectivamente con igual configuración, tal como se muestra en la figura 4b.

4. Conclusiones

En este trabajo se desarrolló una herramienta para la evaluación de las medidas de cercanía e intermediación 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 30.47 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.

Referencias

[1] Mohammed Ali Al-Garadi, Kasturi Dewi Varathan, Sri Devi Ravana, Ejaz Ahmed, and Victor Chang. Identifying the influential spreaders in multilayer interactions of online social networks. *Journal of Intelligent & Fuzzy Systems*, 31(5):2721–2735, 2016-10-13.

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

[3] Laura Calzada-Infante, María Óskarsdóttir, and Bart Baesens. Evaluation of customer behavior with temporal centrality metrics for churn prediction of prepaid contracts. *Expert Systems with Applications*, 160:113553, 2020.

[4] 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, 2019.

[5] 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.

[6] 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, 2017-04-24.

[7] 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.

[8] V. Nicosia F. Battiston and V. Latora. Structural measures for multiplex networks. *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip*, 89(3), 2014.

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

[10] 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.

[11] 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, 2019-08-09.

[12] 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.

[13] 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, 2013.

[14] 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, 2019.

[15] 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.

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

[17] A. Reiffers-Masson and V. Labatut. Opinion-based centrality in multiplex networks: A convex optimization approach. *Netw. Sci*, 5(2):213–234, 2017.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.