Método para decidir los actores más influyentes en una red de desarrolladores

Method for deciding the most influential actors in a network of developers

Eliana Bárbara Ril Valentín [1]

Jorge Alejandro Román Donates [2]

Vladimir Milián Núñez [3]

Raynel Batista Tellez [4]

[1] **Universidad de las Ciencias Informáticas. La Lisa. La Habana. Cuba.**

[2] **Universidad de las Ciencias Informáticas. La Lisa, La Habana. Cuba.**

[3] **Universidad de las Ciencias Informáticas. La Lisa, La Habana, Cuba.**

[4] **Universidad de las Ciencias Informáticas. La Lisa, La Habana, Cuba.**

Resumen

El estudio de las interacciones que establecen los desarrolladores de paquetes en redes de desarrolladores a partir de intereses de desarrollo comunes, contribuye a: identificar sus comunidades, promover la colaboración entre equipos de desarrollo, ayudar a decidir los desarrollos críticos y actores más influyentes (desarrolladores líderes o expertos). El objetivo de esta investigación es desarrollar un método para decidir los actores más influyentes en redes de desarrolladores para fortalecer la colaboración entre equipos de desarrollo. En la investigación se realizó un estudio sobre conceptos asociados a la teoría de grafos, análisis de redes colaborativas y medidas de centralidad. Además, se describió el procedimiento que sigue el método presentado y se realizaron pruebas en aras de verificar la calidad de la solución. Como resultado final se obtuvo un método que facilitó la búsqueda de paquetes en repositorios de sistemas operativos libres, la extracción de los ficheros de control de cambios de cada uno de estos, la extracción de los nom-

bres de paquetes y sus desarrolladores, así como la creación de una red colaborativa a partir de la relación entre desarrolladores y otra red con la relación paquete - desarrollador. El trabajo con Gephi permitió visualizar las redes y los actores más influyentes permitiendo fortalecer la colaboración entre equipos de desarrollo.

Palabras clave: análisis de redes colaborativas, actores más influyentes, desarrolladores, medidas de centralidad.

Abstract

The study of the interactions that package developers establish in networks of developers based on common development interests contributes to: identifying their communities, promoting collaboration between development teams, helping to decide on critical developments and the most influential actors (leading developers or experts). The aim of this research is to develop a method for deciding the most influential actors in networks of developers to strengthen collaboration between development teams. The research included a study of concepts associated with the theory of graphs, analysis of collaborative networks and measures of centrality. In addition, the procedure of the submitted method was described and tests were carried out to verify the quality of the solution. The final result was a method that facilitated the search for packages in free operating system repositories, the extraction of change control files from each of these, the extraction of the names of packages and their developers, as well as the creation of a collaborative network from the relationship between developers and another network with the package - developer relationship. Working with Gephi made it possible to visualize the most influential networks and actors and to strengthen collaboration between development teams, allowing to strengthen the collaboration between development teams.

Keywords: *analysis of collaborative networks, most influential actors, developers, centrality measures.*

Introducción

El estudio de las interacciones que establecen los desarrolladores de paquetes en repositorios de sistemas operativos libres a partir de intereses de desarrollo comunes, contribuye a: identificar sus comunidades, promover la colaboración entre equipos de desarrollo, ayudar a determinar los desarrollos críticos y actores más influyentes (desarrolladores líderes o expertos). Sin embargo, actualmente en los repositorios de los sistemas operativos libres publicados en la UCI se accede manualmente a cada paquete para extraer información de sus desarrolladores. Esto además de resultar arduo por el volumen de información a procesar, eleva el margen de error, involucra más recursos, compromete los resultados esperados, el tiempo de ejecución de los análisis y la precisión de los datos obtenidos. Por lo que, la realización de estos análisis de forma manual pudiera distanciarlos de sus propósitos originales para la toma de decisiones.

Teniendo en cuenta que un repositorio de software libre puede contener miles de paquetes y desarrolladores, como es el caso del repositorio del sistema operativo Nova en la UCI, donde un paquete puede tener uno o varios desarrolladores y un desarrollador colaborar en uno o más paquetes. Según el centro CESOL encargado del desarrollo de Nova, dicha distribución alcanza la cifra de 70 000 paquetes al cierre del 2017 ascendiendo a los 129 Gigabytes, lo cual significa que un repositorio también puede ocupar una alta capacidad de almacenamiento e incrementarse al agregar nuevos paquetes y/o renovar otros, así

como varias versiones de la distribución. Por otra parte, se constató que los repositorios de Ubuntu en la UCI ocupaban 369 Gigabytes en los servidores hospedados en el nodo central al cierre de 2017, lo cual sugirió que la dimensión de las comunidades y el nivel de actividad de sus miembros, ejemplo los desarrolladores de paquetes sea elevada.

Partiendo del contexto anterior se identificó como problema de la presente investigación: ¿Cómo decidir los actores más influyentes en una red de desarrolladores para fortalecer la colaboración entre equipos de desarrollo a través del análisis de redes colaborativas? Delimitado en el objeto de estudio: Análisis de redes colaborativas. Para dar solución al problema planteado, se define como objetivo general: Desarrollar un método para decidir los actores más influyentes en una red de desarrolladores para fortalecer la colaboración entre equipos de desarrollo a través del análisis de redes colaborativas.

Materiales y métodos

Para el desarrollo del método propuesto fue necesario utilizar varios materiales y métodos. En el presente trabajo se decidió usar Python 3.5 para realizar la implementación del método, ya que sobresale en la manipulación de datos y la programación de red por la cantidad de librerías que contiene y que facilitan el trabajo de los desarrolladores (González Duque, 2015). En el trabajo con módulos y librerías se empleó la plataforma Anaconda que además de ser una distribución de código abierto, facilita el proceso de implementar soluciones relacionadas con manipulación de datos en Python. Para representar la red y los resultados que se derivan de esta se utilizó Gephi en su versión 0.9.2 por ser ideal para visualizaciones básicas de la red por la alta calidad que presentan sus resultados. Para el análisis de redes se utilizó la teoría de grafos ya que los grafos son estructuras que constan de dos partes, el conjunto de vértices o nodos y el conjunto de aristas o bordes, que pueden ser orientados o no. Por lo tanto, también es conocida dicha teoría como análisis de redes (Aguirre, 2011; Trudeau, 2013). Además para realizar las pruebas unitarias automáticamente en el lenguaje de programación Python se utilizó el módulo "unittest" al cual se accedió mediante Anaconda3-2.5.0. Por otra parte, los métodos científicos utilizados fueron:

Análisis-síntesis: Facilitó el procesamiento de la información obtenida en la investigación realizada sobre el análisis de redes colaborativas y actores más influyentes. Se realizó un estudio de los conceptos asociados a las redes colaborativas y grafos, así como de las herramientas a utilizar en el desarrollo de la solución propuesta basándose en la revisión de sitios web, trabajos de diploma y artículos científicos.

Modelación: Permitió la creación del modelo conceptual para entender el contexto en el que se enmarca la investigación.

Entrevista: Permitió obtener la información necesaria relacionada con los problemas presentes en el desarrollo de los paquetes del repositorio del sistema operativo Nova.

Resultados y discusión

En la presente investigación los actores más influyentes son desarrolladores que destacan a partir de las relaciones que presentan con respecto a otros dentro de la red colaborativa. El término de red colaborativa es equivalente a redes sociales y se utiliza de esta forma en la investigación para evitar ser confundido con redes como Facebook, Twitter, entre otras. Una red colaborativa se define como una estructura social

compuesta por un conjunto de actores sociales (como individuos u organizaciones) y las relaciones diádicas entre estos actores. La perspectiva de red colaborativa provee un grupo de métodos para analizar la estructura de entidades sociales completas, así como una variedad de teorías explicando los patrones observados en estas estructuras (Wasserman & Faust, 1994). También puede verse como un conjunto de actores vinculados entre sí (Monsalve Moreno, 2008). Para modelar y analizar redes colaborativas se utilizan los grafos. En este contexto los vértices representan a los actores (autores, desarrolladores, etc) y las aristas representan las relacionen existentes entre estos.

En una red colaborativa los actores más influyentes pueden identificarse a través de las medidas de centralidad. *"Las medidas de centralidad son métricas fundamentales para el análisis de redes. Miden cómo de central e importante es un nodo dentro de la red"* (Lozano, García-Martínez, Rodríguez, & Trujillo, 2016). En la investigación se trabajaron las medidas de centralidad de grado, cercanía e intermediación. La centralidad de grado asume que los nodos más importantes son los que tienen muchas conexiones con otros nodos. "Los investigadores de redes sociales miden la actividad en la red usando el concepto de centralidad de grado, es decir el número de conexiones directas que tiene un nodo"(Kuz, Falco, & Giandini, 2016). Consiste en nodos que, independientemente de la cantidad de conexiones, sus aristas permiten llegar a todos los nodos de la red más rápidamente que desde cualquier otro nodo.

La cercanía se basa en la medida de proximidad y en su opuesta, la lejanía. Describe mejor la centralidad general, ya que los actores (nodos) son valorados por su distancia, medida en pasos hacia los demás actores de la red. Un actor tiene gran centralidad cuanto menor sea el número de pasos que a través de la red debe realizar para relacionarse con el resto (Beltrán, Eduardo, Valerio Ureña, & Rodríguez-Aceves, 2015; Kuz et al., 2016). La centralidad de cercanía asume que los nodos importantes son aquellos que están a una corta distancia del resto de los nodos de la red.

Por último se trabajó con la centralidad de intermediación que mide la importancia de un elemento de un grafo, ya sea un nodo o una arista, por la fracción de los caminos más cortos que pasan a través de él (Kourtellis, Morales, & Bonchi, 2014). La centralidad de intermediación es una medida de centralidad muy popular que, informalmente, define la importancia de un nodo o borde en la red como proporcional a la fracción de rutas más cortas en la red que pasan por él (Riondato & Upfal, 2016). La centralidad de intermediación asume que los nodos importantes en la red son aquellos que conectan otros nodos.

Para el cumplimento del objetivo general del presente trabajo se deben decidir los actores más influyentes en una red colaborativa donde se representan los paquetes del repositorio de los sistemas operativos libres y su relación con los desarrolladores que intervienen en cada uno de ellos. El método propuesto sigue el procedimiento que se describe a continuación:

1. Se extraen los ficheros de control de cambios o changelogs de los paquetes que se encuentran en el repositorio.

2. Se extraen los datos útiles a partir de los ficheros de control de cambios obtenidos del paso anterior, con un algoritmo de extracción de datos que propone el autor del presente trabajo, para seleccionar los nombres de los paquetes y los desarrolladores que intervienen en cada uno.

3. Se crea la red colaborativa a partir de los datos obtenidos.

4. Se detectan los actores más influyentes en la red.

Un fichero de control de cambios o changelog es un archivo en el que se encuentra la información en orden cronológico sobre los cambios realizados en cualquier proyecto de tipo informático. El objetivo del changelog es mostrarle a usuarios y desarrolladores los cambios que sean implementados en las diferentes versiones del producto, así como la información acerca de quien realizó dichos cambios para poder contactarlo si se detectan errores o fallas de seguridad. Los ficheros tienen similar estructura lo cual permite que la salida sea siempre similar independientemente de la cantidad de entradas que se procesen.

Los ficheros de control de cambios se pueden obtener descomprimiendo los paquetes en formato ".deb" que se encuentran en el repositorio de las distribuciones GNU/Linux. Para acceder a los repositorios es necesario conocer las direcciones de los mismos las cuales suelen ser URL's (Localizador Uniforme de Recursos, por sus siglas en español), las empresas encargadas de desarrollar distribuciones GNU/Linux tienen públicas estas direcciones, al tratarse de software libre pueden existir también repositorios locales propios de organizaciones o instituciones con intereses específicos.

En el desarrollo de la propuesta de solución se trabajó primeramente con el módulo "os" cuyas funciones permiten el trabajo con directorios de carpetas y archivos. En el presente trabajo se empleó para crear directorios de carpetas con los que trabaja el método para descargar paquetes del repositorio, copiar los changelogs y generar los ficheros o archivos con las redes colaborativas. Para el trabajo con las URL se utilizaron varias librerías como "urllib.request" para acceder a estas y descargar los paquetes hacia directorios previamente creados de forma automática. Otra librería que se empleó para trabajar con las URL fue "urllib3" para obtener direcciones con las que pudiera tratar la librería "BeautifulSoup", esta última permitió obtener de una página web todos los enlaces que la componen y de esta forma iterar recursivamente por todo el repositorio de paquetes y lograr la descarga de cada archivo.

También se emplearon módulos como "patoollib" para desempaquetar o extraer archivos ".deb" y dar paso a la utilización de las librerías "shutil" y "gzip" para la descompresión de archivos. Los archivos descomprimidos fueron el archivo "data.tar.xz" y "changelog.Debian.gz" con el objetivo de acceder el fichero de control de cambios de cada paquete, de igual forma se eliminaron los paquetes después de ser extraídos para ahorrar espacio en el disco duro de la computadora donde se ejecute el método.

Para la extracción de los changelogs se propone el pseudocódigo siguiente:
Entrada del algoritmo: Dirección URL de un repositorio de sistema operativo de software libre.

Salida del algoritmo: Directorio con los ficheros de control de cambios de los paquetes del repositorio.

1:	**Si** URL == paquete
2:	**Llamar** método ddcr
3:	**Fin**
4:	**Sino**
5:	**Obtener** direcciones URL
6:	**Para** direcciones URL hacer

7:	**Si** dirección url == "../"
8:	**Volver** a ejecutar el método
9:	**Fin**
10:	**Fin**
11:	**Fin**

El método ddcr es el encargado de descargar los paquetes del repositorio y para cada uno de estos realizar el proceso de desempaquetado, descompresión de los ficheros comprimidos y extracción del changelogs. Este método además copia los changelogs hacia un directorio que sirve como entrada para el proceso de extracción de datos útiles.

A partir de los ficheros de control de cambios obtenidos de una dirección dentro del sistema operativo como "D:\prueba" (si se trabaja sobre cualquier distribución de Windows) se extrajeron los datos útiles, para posteriormente adicionarlos a una lista. En la presente investigación se definen como datos útiles:

- nombres de los paquetes. Ejemplo: "firefox (46.0+build5-0ubuntu0.14.04.2nova1)".
- nombres de los desarrolladores de los paquetes. Ejemplo: "Juan Manuel Fuentes Rodríguez", "Chris Coulson", "Alexander Sack".

Python permite cargar archivos de texto usando para ello la función "open" a la que se le pasan como parámetros el archivo o ruta del mismo y el modo en que se desea cargar el archivo (lectura, escritura, etc), la carga de los archivos es necesaria para poder realizar operaciones sobre el texto. Una vez cargado el texto, se procede a extraer los datos útiles, para esto es necesario trabajar con expresiones regulares y puede emplearse el módulo "Re" el cual contiene varias funciones para simplificar el código en este sentido.

En la solución se realizó primeramente un filtro de las oraciones que contengan algún correo electrónico teniendo en cuenta que estos tienen como singularidad la existencia del carácter "@". La estructura definida de los changelogs facilita el uso de expresiones regulares aprovechando que los nombres de desarrolladores se encuentran en oraciones donde aparece al menos un correo electrónico. Posteriormente se definió una expresión regular para identificar los datos útiles, para los nombres de desarrolladores la expresión es "[A-Z][a-z]+ [A-Z][a-z]". Para identificar los nombres de paquetes también se definió una expresión regular pero no es necesario filtrar el texto ya que este dato se encuentra al inicio de cada archivo, en la solución propuesta se utiliza la expresión "\A\w+".

Se propone el siguiente pseudocódigo para la extracción de datos útiles:

Entrada del algoritmo: Directorio de changelogs.
Salida del algoritmo: Lista con nombres de paquetes y desarrolladores de cada paquete.
Aclaraciones: Se debe indicar la dirección donde se encuentran los ficheros de control de cambios al algoritmo propuesto.

1:	**Para** cada fichero hacer
2:	**Para** cada oración hacer
3:	**Si** carácter == "@"
4:	**Si** expresión regular desarrollador == palabras
5:	**Agregar** palabras a la lista de desarrolladores de un paquete
6:	**Fin**
7:	**Fin**
8:	**Agregar** lista de desarrolladores de un paquete a lista de todos los desarrolladores
9:	**Fin**
10:	**Si** expresión regular paquete== palabras
11:	**Agregar** palabras a la lista de paquetes
12:	**Fin** el nombre del paquete
13:	**Fin**
14:	**Para** longitud de lista de todos los desarrolladores hacer
15:	**Agregar** lista de todos los desarrolladores + lista de paquetes a lista general
16:	**Fin**

Para crear la red colaborativa a partir de la información obtenida en el proceso de extracción de datos útiles descrito anteriormente. Los datos útiles procedentes del proceso anterior se obtienen como una lista de listas donde el primer elemento de cada lista es el nombre del paquete y los restantes elementos son el nombre de los desarrolladores asociados a este.

Para el trabajo con grafos en Python se hace uso de la librería "NetworkX" con el objetivo de simplificar el trabajo y garantizar una buena calidad en la implementación. Esta librería permitió diseñar, crear, probar y exportar la red colaborativa a través del trabajo con un conjunto de funciones implementadas en "NetworkX". Se crearon dos redes, en la primera los nodos o vértices fueron cada nombre de paquete y desarrollador, donde las aristas que conectan dichos nodos establecen la relación entre estos según se obtienen los datos en la lista de entrada. La segunda red establece como nodos solamente a los desarrolladores siendo los paquetes asociados a estos los datos que permiten establecer las relaciones en la red mediante aristas. La librería en cuestión exporta las redes colaborativas en varios formatos como por ejemplo "gml" creándose de esta forma dos archivos de este tipo para poder ser posteriormente trabajados con Gephi.

Para crear la red colaborativa el programador puede auxiliarse del siguiente pseudocódigo propuesto por los autores de la investigación:

Entrada del algoritmo: Lista de nombres de paquetes y desarrolladores.

Salida del algoritmo: Fichero ".gml" con la red colaborativa.

1:	**Para** lista de todos los desarrolladores hacer
2:	**Agregar** desarrolladores a la lista de nodos de la bipartición 0
3:	**Fin**
4:	**Para** lista de paquetes hacer
5:	**Agregar** paquetes a la lista de nodos de la bipartición 1
6:	**Fin**
7:	**Para** A en lista de todos los desarrolladores hacer
8:	**Para** B en A hacer
9:	**Agregar** la relación de paquetes a desarrolladores a la lista de aristas
10:	**Fin**
11:	**Fin**
12:	**Crear** archivo "gml" para la red paquete-desarrollador
13:	**Para** longitud de lista de todos los desarrolladores hacer
14:	**Agregar** desarrolladores a la lista de nodos con el atributo paquete
15:	**Fin**
16:	**Para** A en lista de todos los desarrolladores hacer
17:	**Para** B en longitud de A hacer
18:	**Para** C en A hacer
19:	**Agregar** la relación entre desarrolladores a la lista de aristas
20:	**Fin**
21:	**Fin**
22:	**Fin**
23:	**Crear** archivo "gml" para la red de desarrolladores

El análisis de la red colaborativa se realiza con la herramienta profesional para la visualización de redes Gephi. El análisis con dicha herramienta permite observar con mayor detalle las relaciones que se presentan en la red colaborativa, moverse por los nodos o vértices con mayor facilidad, así como ajustar la forma en la que se presentan estos sin dañar las relaciones. Gephi permite la detección de actores más influyentes a partir de las medidas de centralidad previamente mencionadas, como se muestra en la siguiente figura:

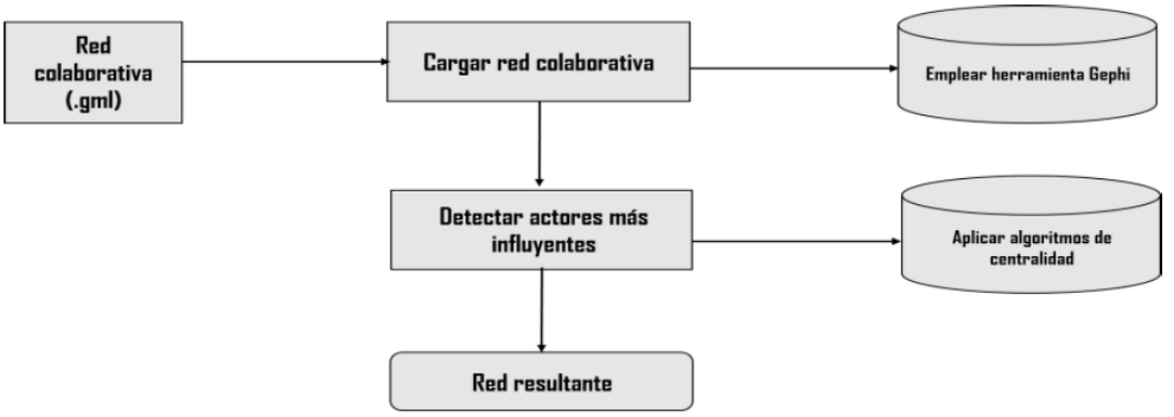

Figura 1. Esquema de detección de actores más influyentes en la red colaborativa (Fuente: Elaboración propia)

Basándose en la medida de centralidad se pudieron identificar los actores más influyentes en la red. La siguiente figura muestra cómo quedaría la red al aplicarle la medida de intermediación que establece que los nodos más importantes son aquellos que conectan otros nodos.

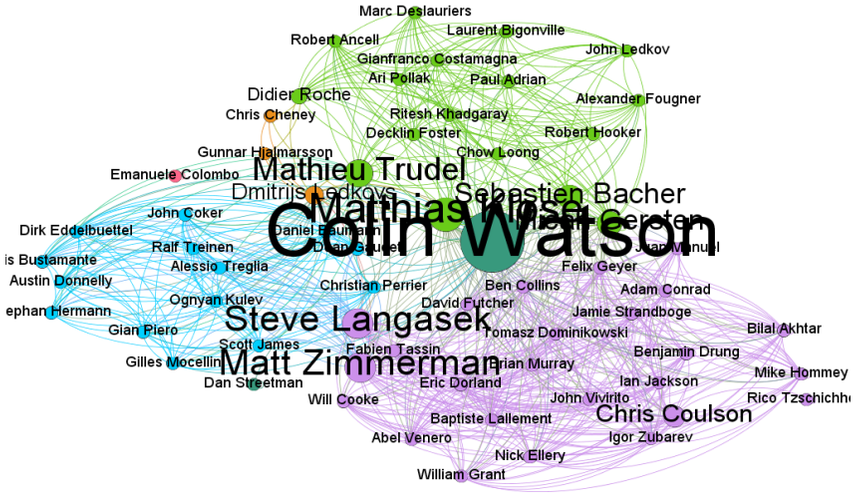

Figura 2. Desarrolladores más importantes de la red colaborativa (Fuente: Elaboración propia)

Una vez obtenido el método propuesto se realizaron varias pruebas al código para asegurar la calidad y verificar que no existan fallas mediante pruebas unitarias. Además, se detectaron un grupo de no conformidades y recomendaciones después de realizar 3 iteraciones de pruebas de aceptación, las cuales fueron corregidas.

Conclusiones

Al concluir la presente investigación se pueden arribar a las siguientes conclusiones que dan solución al objetivo general de la misma:

1. Las medidas de centralidad permitieron identificar los actores más influyentes dentro de la red colaborativa.

2. Se obtuvo un método implementado en Python que facilitó la búsqueda de paquetes en repositorios de sistemas operativos libres y la extracción de los ficheros de control de cambios de cada uno de estos.

3. El método implementado facilitó la extracción de los nombres de paquetes y sus desarrolladores, así como la creación de una red colaborativa a partir de la relación entre desarrolladores y otra red con la relación paquete - desarrollador.

4. A partir del trabajo con la herramienta Gephi se pudieron visualizar las redes colaborativas y proceder a la detección de los actores más influyentes en la red permitiendo fortalecer la colaboración entre equipos de desarrollo.

5. La validación realizada a través de pruebas de software permitió corregir las no conformidades detectadas, probando la calidad de la solución propuesta.

Referencias

Aguirre, J.-L. (2011). Introducción al análisis de redes sociales. *Documentos de Trabajo del Centro Interdisciplinario para el Estudio de Políticas Públicas*, *82*, 1-59.

Beltrán, P., Eduardo, J., Valerio Ureña, G., & Rodríguez-Aceves, L. (2015). Análisis de redes sociales para el estudio de la producción intelectual en grupos de investigación. *Perfiles educativos*, *37*(150), 124-142.

González Duque, R. (2015). *Python para todos*. España. Recuperado a partir de http://mundogeek.net/tutorial-python/

Gross, J. L., Yellen, J., & Zhang, P. (2013). *Handbook of graph theory* (2da ed.). Chapman and Hall/CRC.

Kourtellis, N., Morales, G. D. F., & Bonchi, F. (2014). Scalable Online Betweenness Centrality in Evolving Graphs. *arXiv:1401.6981 [cs]*. Recuperado a partir de http://arxiv.org/abs/1401.6981

Kuz, A., Falco, M., & Giandini, R. (2016). Análisis de redes sociales: un caso práctico. *Computación y Sistemas*, *20*(1), 89-106.

Lozano, M., García-Martínez, C., Rodríguez, F. J., & Trujillo, H. M. (2016). Optimización de ataques a redes complejas mediante un algoritmo de colonias de abejas artificiales. Recuperado a partir de http://helvia.uco.es/bitstream/handle/10396/15775/articulo-Actas-maeb2016.pdf?sequence=1&isAllowed=y

Monsalve Moreno, M. (2008). Análisis de Redes Sociales: un Tutorial, 6.

Riondato, M., & Upfal, E. (2016). ABRA: Approximating betweenness centrality in static and dynamic

graphs with rademacher averages. En *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1145-1154). ACM.

Trudeau, R. J. (2013). *Introduction to graph theory*. Courier Corporation.

Wasserman, S., & Faust, K. (1994). *Social Network Analysis: Methods and Applications* (1ra ed.). Reino Unido: Cambridge University Press. Recuperado a partir de http://www.google.com.cu/books?hl=en&lr=&id=CAm2DpIqRUIC&oi=fnd&pg=PR21&dq=stanley+wasserman&ots=HvHpA-fWIR9&sig=XZaTkWrq0ehCeond9h9bz1VE2C8&redir_esc=y#v=onepage&q&f=false