Un Algoritmo Memético, con búsqueda local basada en Label Propagation, para detectar comunidades en redes dinámicas

1st Ángel Panizo
Departamento de ciencias de la computación
Universidad Autónoma de Madrid
Madrid, España
angel.panizo@uam.es

2nd Gema Bello-Orgaz
Departamento de ciencias de la computación
Universidad Autónoma de Madrid
Madrid, España
gema.bello@uam.es

3rd Alfonso Ortega
Departamento de ciencias de la computación
Universidad Autónoma de Madrid
Madrid, España
alfonso.ortega@uam.es

4th David Camacho
Departamento de ciencias de la computación
Universidad Autónoma de Madrid
Madrid, España
david.camacho@uam.es

***Abstract*—El análisis y la detección de comunidades en redes complejas es actualmente un área de estudio en auge, ya que muchos sistemas se pueden representar como redes de nodos interconectados. Tradicionalmente el esfuerzo se ha aplicado en estudiar métodos para analizar redes estáticas, es decir, redes que no cambian en el tiempo. En el mundo real, a menudo, las redes son dinámicas, es decir, evolucionan según pasa el tiempo, debido a este fenómeno la comunidad científica se ha comenzado a interesar por el análisis de este tipo de redes. En la literatura ya se han usado con éxito tanto algoritmos genéticos como otros algoritmos bio-inspirados para detectar comunidades en redes dinámicas, pero pocos artículos se han centrado en buscar mecanismos para mejorar estos métodos explotando la temporalidad de estas redes. En este artículo presentamos un Algoritmo Memético, con búsqueda local basada en *Label Propagation*, para identificar comunidades en redes dinámicas.**

***Index Terms*—detección dinámica de comunidades, algoritmos genéticos, análisis de redes complejas, algoritmos meméticos**

I. Introducción

El análisis y la detección de comunidades en redes complejas es actualmente un área de estudio en auge, ya que muchos sistemas complejos se pueden representar como redes de nodos interconectados. Esta idea se ha aplicado de forma satisfactoria a una gran cantidad áreas de estudio: como el marketing [1], la salud pública [2], el cibercrimen [3]–[5] o el análisis de redes sociales [6]–[8]. La detección de comunidades es un problema *mal definido*, ya que existen varias definiciones de qué es una comunidad y no se ha llegado a un consenso entre la comunidad científica para decidir cuál de ellas es la correcta. Nosotros definiremos una comunidad como un conjunto de nodos que interactúan más entre ellos que entre el resto de nodos de la red. Además, podemos definir dos tipos de comunidades, *solapadas* y *no-solapadas*, dependiendo del número de comunidades a las que un nodo puede pertenecer a la vez. Si un nodo puede pertenecer a varias comunidades a la vez, entonces estaremos hablando de comunidades *solapadas*. Mientras que si un nodo sólo puede pertenecer a una única comunidad, entonces hablaremos de comunidades *no-solapadas*.

Tradicionalmente el esfuerzo se ha aplicado en estudiar métodos para analizar redes estáticas, es decir, redes que no cambian en el tiempo. En el mundo real, a menudo, las redes son dinámicas, es decir, evolucionan según pasa el tiempo. Debido a este fenómeno la comunidad científica se ha comenzado a interesar por el análisis de este tipo de redes. Una red dinámica se puede modelar usando una secuencia de instancias, dónde cada instancia representa el estado de la red en un momento concreto. Las comunidades de una red dinámica son conocidas como *comunidades dinámicas*. Detectar comunidades dinámicas tiene una dificultad añadida si lo comparamos con sus homologas estáticas. No sólo es necesario agrupar cada instancia de la red en comunidades, sino que también hay que rastrear estas comunidades a lo largo de las diferentes instancias de la red.

Los métodos de detección de comunidades dinámicas se pueden dividir en dos grupos. El primer grupo incluye a aquellos métodos que hacen el paso de agrupación y el paso de rastreo de manera independiente, por ejemplo, primero se agrupa cada instancia de la red en comunidades y después se analizan los cambios que ha sufrido estas comunidades a lo largo de las distintas instancias. El segundo grupo, incluye los métodos que siguen la filosofía del *evolutionary clustering* [9], que consiste en agrupar cada instancia de la

Este trabajo ha sido cofinanciado por los siguientes proyectos de investigación: EphemeCH (TIN2014-56494-C4-4-P) y DeepBio (TIN2017-85727-C4-3-P). Ministerio de Economía y Competitividad de España, bajo los fondos Europeos de desarrollo regional (FEDER).

red de tal manera que las comunidades encontradas en una instancia se parezcan, en la mayor medida de lo posible, a las comunidades encontradas en la instancia anterior. Esta filosofía asume que no se dan cambios abruptos en la red en un periodo corto de tiempo. Por motivos de claridad, a partir de ahora, a los métodos del primer grupo los llamaremos *métodos temporalmente suaves* y a los métodos del segundo grupo *métodos temporalmente no-suaves*.

Los Algoritmos Genéticos (GAs) son una meta-heurística inspirada en los principio de la selección natural de Charles Darwin y en la genética mendeliana. Los GAs son un método efectivo de resolver problemas de optimización combinatoria y se han usado anteriormente para detectar comunidades tanto estáticas [10], [11] como dinámicas [12]. También se han usado Algoritmos Meméticos para detectar comunidades estáticas. Un Algoritmo Memético es un Algoritmo genético que se ha emparejado con un método de *búsqueda local*. Los métodos de *búsqueda local* se han aplicado principalmente durante la creación de la población inicial [13] o durante el proceso de mutación [14]. En este artículo presentamos un Algoritmo Memético que usa un proceso de *búsqueda local* basado en *Label Propagation* [15] para detectar comunidades dinámicas, *no solapadas* y siguiendo un *método temporalmente no-suave*. En el método de *Label Propagation* cada nodo, de manera iterativa, cambia su comunidad para pertenecer a la misma comunidad que la mayoría de sus vecinos. El algoritmo termina cuando en una iteración ningún nodo ha cambiado su comunidad. La idea principal de nuestro algoritmo es utilizar las comunidades detectadas en una instancia de la red para guiar la búsqueda de la instancia siguiente. Nuestro método ejecuta un GAs para cada instancia de la red dinámica. La población de cada GAs se construye utilizando la población obtenida como solución para la instancia anterior. Debido a los cambios en la red, es posible que algunos de los individuos dejen de ser soluciones válidas cuando se pasan de una instancia a otra. Para solucionar este problema proponemos usar una búsqueda local basada en *Label Propagation* para *reparar* estos individuos intentando mantener la mayor cantidad de información posible. Este trabajo se centra en la agrupación de las diferentes instancias de tiempo en comunidades, dejando el posterior rastreo fuera del ámbito de este.

La principal contribución de este trabajo es la presentación y el análisis de la viabilidad de un método de búsqueda local para mejorar la eficiencia de un GAs capaz de detectar comunidades dinámicas.

El resto del artículo está organizado de la siguiente manera: en la sección II se describe la detección dinámica de comunidades de una manera formal y se presenta el Algoritmo Memético propuesto. En la sección III se presentan los procedimientos seguidos para comprobar la viabilidad y la eficiencia del método usando un *dataset* real. Por último, se muestran las conclusiones obtenidas y se proponen futuras líneas de investigación.

II. Algoritmo

A. Formulación del problema

Sea $\aleph = \{G^0..G^n\}$ una red dinámica con n instancias ordenadas en orden cronológico. Sea G^t una instancia de dicha red modelada como un grafo $G^t(V^t, E^t)$. Sea V^t el conjunto de vértices o nodos presentes en el instante t. Sea E^t un conjunto de conexiones, llamadas aristas, que conectan dos elementos de V^t en un instante concreto de tiempo t. Nuestro objetivo es agrupar los vértices de cada instancia de la red dinámica G^t de tal manera que los vértices que pertenezcan a un mismo grupo tengan más conexiones entre ellos que con el resto de vértices de G^t y además cada vértice pertenezca a un único grupo en cada instancia de tiempo.

B. Algoritmo Memético

El pseudocódigo del Algoritmo Memético propuesto está disponible en el *Algoritmo* 1. Usamos un Algoritmo Genético con elitismo para detectar las comunidades en cada instancia, G^t, de la red dinámica $\aleph$ (líneas 1-13). Para la primera instancia de la red, G^0, el método evoluciona una población generada aleatoriamente. Sin embargo, para el resto de las instancias, $G^t|t > 0$, en vez de evolucionar una población aleatoria, se evoluciona una población generada a partir de la población obtenida como solución para la instancia anterior. Una vez procesadas todas las instancias G^t el resultado devuelto por el algoritmo ($C_{dinamica}$) es la solución codificada por el mejor individuo de cada una poblaciones evolucionadas (línea 13).

Cada individuo de una población codifica una posible agrupación de nodos en comunidades para una instancia G^t de la red. Los individuos codifican las soluciones usando la codificación *locus-based adjacency* propuesta en [16]. Tal como comentamos en el párrafo anterior, para la primera instancia de la red, G^0, generamos una población aleatoria (línea 7). Para generar cada individuo de la población aleatoria rellenamos cada uno de sus genes con un vecino aleatorio del nodo que codifica dicho gen. Por el contrario, para el resto de instancias de la red ($\{G^t|t > 0\}$), los individuos de la población inicial se generan aplicando la función *labelBL* a cada uno de los individuos devueltos por el Algoritmo Genético ejecutado para la instancia anterior (lines 9-11). La función *labelBL* es la función de búsqueda local basada en *Label Propagation* que *traspasa* la información de un individuo de una generación a otra. Se encarga de cambiar los genes de un individuo que remiten a nodos que han dejado de ser vecinos de el nodo que el gen codifica (líneas 16-23). Para seleccionar un nuevo valor para cada uno de estos genes: primero buscamos todos los vecinos del nodo, que codifica el gen afectado, en la instancia actual G^t (línea 18). A continuación, utilizamos la mejor solución (C_{last}) de la instancia anterior G^{t-1} para encontrar la comunidad a la que pertenecen el mayor número de vecinos (línea 20). Por último, el valor del gen se cambia por un vecino aleatorio que pertenezca a la comunidad seleccionada en el paso anterior (línea 22).

Una vez generada la población inicial, se evoluciona siguiendo un Algoritmo Genético Elistista (línea 12) con las siguientes funciones: *Two Points Crossover*, *Tournament Selection* y *Modularity* [17] como *función de Fitness*. Además, como mutación se ha implementado una función que selecciona una serie de genes al azar y cambia sus valores por vecinos aleatorios del nodo que codifica el gen afectado. Estas funciones se seleccionaron después de revisar los operadores más utilizados en la literatura.

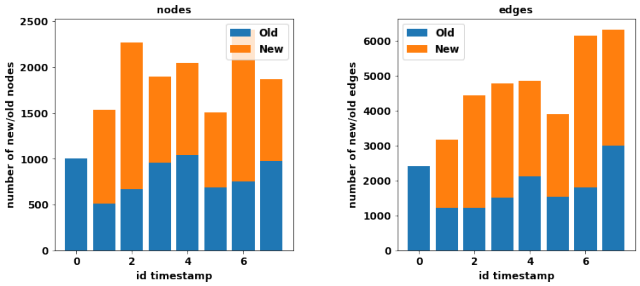
Algorithm 1 Algoritmo Memético para detectar comunidades dinámicas

1: **function** AlgoritmoMemetico($\aleph$)
2: $Conf \leftarrow ELITISM, CRXRATE$
3: $Conf \leftarrow Conf \cup MUTR, MAXGEN, NCONV$
4: $C_{dinamica} \leftarrow \emptyset$
5: $P_{ant} \leftarrow null$
6: $C_{ant} \leftarrow null$
7: **for all** $G^t \in \aleph$ **do**
8: **if** $t = 0$ **then**
9: $P_{ini} \leftarrow poblacionAleatoria(G^t, POPSIZE)$
10: **else**
11: $P_{ini} \leftarrow \emptyset$
12: **for all** $ind \in P_{ant}$ **do**
13: $P_{ini} \leftarrow P_{ini} \cup labelBL(ind, G^t, C_{ant})$
14: $P_{ant} \leftarrow elitistaGA(P_{ini}, G^t, Conf)$
15: $C_{ant} \leftarrow mejorSolucion(P_{ant})$
16: $C_{dinamica} \leftarrow C_{dinamica} \cup C_{ant}$
17: **return** $C_{dinamica}$
18: **function** labelBL(ind, G^t, C_{ant})
19: **for all** $i \in [1, size(ind)]$ **do**
20: $vecinos \leftarrow getVecinos(i, G^t)$
21: **if** $ind[i] \notin vecinos$ **then**
22: $com \leftarrow getComunidadComun(vecinos, C_{ant})$
23: $vecinosOk \leftarrow nodosEnCom(vecinos, com)$
24: $ind[i] \leftarrow seleccionAzar(vecinosOk)$
25: **return** $individuo$

III. Experimentación

A. Descripción del dataset

En esta sección estudiaremos la efectividad del algoritmo propuesto probándolo contra el *dataset* de *Enron* [18]. Este dataset consta de los correos enviados por los empleados de la corporación Enron entre los años de 1999 y 2003. Cada vértice en el dataset representa a un trabajador de Enron y una artista uniendo dos nodos significa que esos dos empleados mantuvieron una comunicación por e-mail. Cada arista tiene asociada la fecha de cuando se produjo dicha comunicación. Usando estas fechas se ha creado una red dinámica compuesta por ocho instancias. Cada instancia tiene los correos enviados entre empleados de Enron durante quince días, los fines de semana se han excluido debido a la poca actividad que ocurre en ellos. Se han seleccionado quince días como la resolución de cada instancia de la red porque una resolución menor daba lugar a redes demasiado dinámicas para ser analizadas usando este método. Las ocho instancias que se han seleccionado son consecutivas y empiezan el 1 de Enero del 2000 y terminan el 30 de Abril del mismo año. En la figura 1 se muestran el número de nodos y aristas de cada instancia.



(a) (b)

Fig. 1: Número de vértices y aristas en cada instancia. La parte azul de la barra muestra el número de elementos que estaban presentes en la instancia anterior además de en la actual. La parte naranja muestra el número de elementos que están presentes en la estancia actual pero no en la anterior.

TABLE I: Parámetros utilizados en los Algoritmos Genético y Memético

Parámetro	Descripción	Valor
$POPSIZE$	Tamaño de la población	300
$ELITISM$	Número de individuos que forman la élite	10%
$MAXGEN$	Número de máximo de generaciones	500
$NCONV$	Número de generaciones con el mismo *fitness* (+/-0.01)	10
$CRXRATE$	Probabilidad de cruce	1.0
$MUTR$	Probabilidad de Mutación	0.1

B. Resultados Experimentales

Para poder validar nuestra propuesta vamos a comparar nuestro Algoritmo Memético contra un Algoritmo Genético que no usa búsqueda local (para cada instancia de la red evoluciona una población aleatoria). Para evaluar ambos algoritmos compararemos el *fitness* del mejor individuo de cada generación obtenidos por los dos métodos. De esta manera no sólo podremos comparar cuál de los dos algoritmos obtiene un resultado mejor, sino que también podremos valorar cuál de ello lo obtiene en menos generaciones. Tanto el Algoritmo Memético como el Genético usan la misma configuración que se encuentra disponible en la Tabla I. Los diferentes parámetros se han calibrado de manera manual siguiendo un paradigma de ensayo y error.

Cada experimento se ha ejecutado 30 veces, la mediana de todas las ejecuciones está disponible en las Figuras 2 y 3. Analizando los resultados podemos concluir que nuestro Algoritmo Memético consigue mejores resultados en todas las instancias de la red a excepción de la *instancia 3* (Figura 3a). La primera instancia, la cero (Figura 2a), es un caso especial dado que al no existir una solución previa no se puede

llevar a cabo la búsqueda local y se comporta igual que el Algoritmo Genético, obteniendo ambos resultados similares. Para el resto de instancias, a excepción de la tres, la población obtenida por el Algoritmo Memético es mejor que la población obtenida por el Genético durante toda la ejecución. En la *instancia 3* la población generada por el Mémetico comienza siendo mejor que la población del Genético, pero en la generación 15 este le sobrepasa y se mantiene así hasta el final. En nuestra opinión aquí yace el principal problema de este método: aunque las poblaciones del Memético siempre empiezan siendo mejores que la del genético, la velocidad de convergencia de las primeras son menores que la velocidad de las segundas. Esto se debe a que el método de búsqueda local, usado por el Algoritmo Memético , genera poblaciones con una diversidad menor que el método aleatorio, usado por el Genético, reduciendo las capacidades exploratorias del primero. Este efecto hace que el Algoritmo Memético sea más propenso a quedarse atascado en un mínimo local, justo lo que está pasando en la *instancia 3*.

IV. Conclusiones

En este artículo hemos presentado un nuevo Algoritmo Memético para detectar comunidades en redes dinámicas. El método ejecuta un Algoritmo Genético elitista para cada una de las instancias que conforman la red dinámica. Entre cada ejecución del Algoritmo Genético se aplica un operador de búsqueda local basado en *Label propagation* que aprovecha la solución anterior para guiar la ejecución actual y obtener mejores resultados. Para probar la validez del método se ha comparado nuestro algoritmo contra un Algoritmo Genético sin búsqueda local usando el *dataset* de Enron. Los experimentos que se han llevado a cabo muestran que el Algoritmo Memético encuentra soluciones de más calidad que el Genético, sin embargo, también muestran que el Algoritmo Memético tiene menos capacidad exploratoria que el Genético y tiene más posibilidades de quedar estancado en un mínimo local, aunque en la mayoría de los casos estudiados esto no ocurrió. Por está razón, nuestras futuras líneas de investigación se centrarán en estudiar otros métodos de búsqueda local o combinaciones de estos que generen poblaciones que sean a su vez diversas y de calidad. Además, también compararemos nuestro método con otros métodos del estado del arte y otros *dataset*, tanto sintéticos como reales, para determinar con más certeza si el método propuesto es o no prometedor.

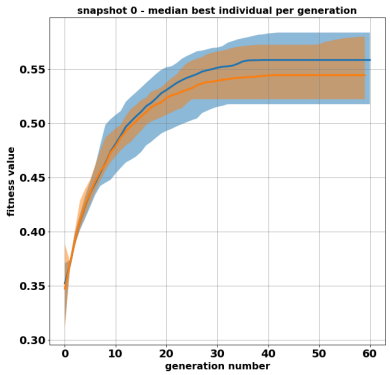
References

[1] G. Bello-Orgaz, H. Menéndez, S. Okazaki, and D. Camacho, "Combining social-based data mining techniques to extract collective trends from twitter," *Malaysian Journal of Computer Science*, vol. 27, no. 2, 2014.

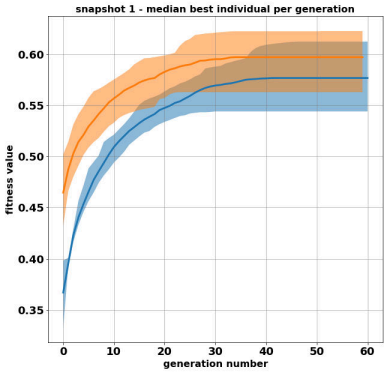
[2] G. Bello-Orgaz, J. Hernandez-Castro, and D. Camacho, "Detecting discussion communities on vaccination in twitter," *Future Generation Computer Systems*, vol. 66, pp. 125–136, 2016.

[3] R. Lara-Cabrera, A. Gonzalez-Pardo, M. Barhamgi, and D. Camacho, "Extracting radicalisation behavioural patterns from social network data," in *2017 28th International Workshop on Database and Expert Systems Applications (DEXA)*, Aug 2017, pp. 6–10.

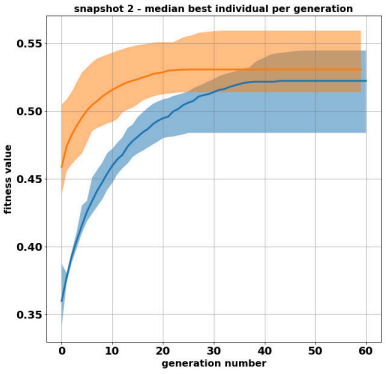
[4] R. Lara-Cabrera, A. Gonzalez-Pardo, and D. Camacho, "Statistical analysis of risk assessment factors and metrics to evaluate radicalisation in twitter," *Future Generation Computer Systems*, 2017.



(a)

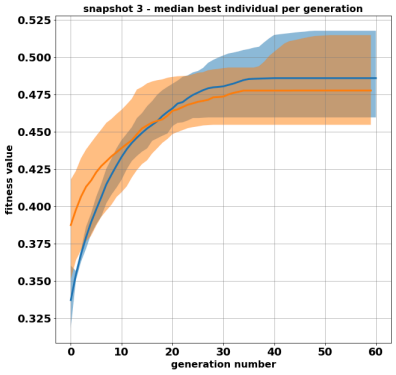


(b)

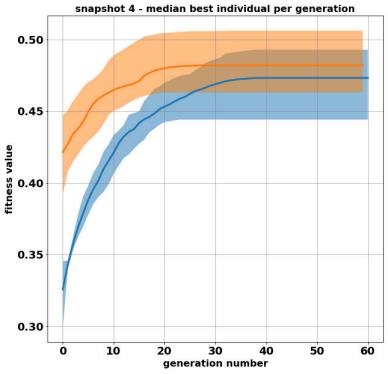


(c)

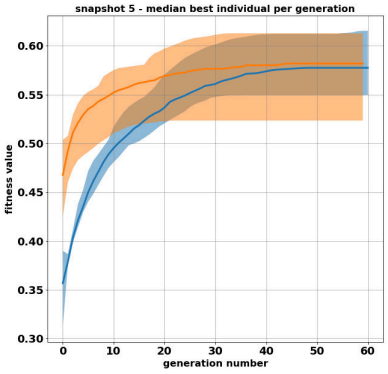
Fig. 2: Cada gráfico corresponde con una instancia de la red. En el eje X se muestra el número de generación. En eje Y el valor de la modularidad de el mejor individuo de la población. La curva naranja muestra la evolución de las poblaciones generadas pos el Algoritmo Memético y la azul la evolución del Algoritmo Genético.



(a)

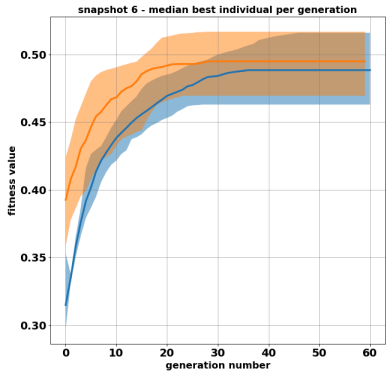


(b)

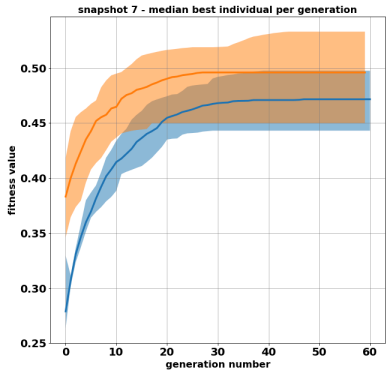


(c)

Fig. 3: Cada gráfico corresponde con una instancia de la red. En el eje X se muestra el número de generación. En eje Y el valor de la modularidad de el mejor individuo de la población. La curva naranja muestra la evolución de las poblaciones generadas pos el Algoritmo Memético y la azul la evolución del Algoritmo Genético.



(a)



(b)

Fig. 4: Cada gráfico corresponde con una instancia de la red. En el eje X se muestra el número de generación. En eje Y el valor de la modularidad de el mejor individuo de la población. La curva naranja muestra la evolución de las poblaciones generadas pos el Algoritmo Memético y la azul la evolución del Algoritmo Genético.

[5] A. Malm and G. Bichler, "Networks of collaborating criminals: Assessing the structural vulnerability of drug markets," *Journal of Research in Crime and Delinquency*, vol. 48, no. 2, pp. 271–297, 2011.

[6] A. Gonzalez-Pardo, J. J. Jung, and D. Camacho, "Aco-based clustering for ego network analysis," *Future Generation Computer Systems*, vol. 66, pp. 160–170, 2017.

[7] R. Lara-Cabrera, A. G. Pardo, K. Benouaret, N. Faci, D. Benslimane, and D. Camacho, "Measuring the radicalisation risk in social networks," *IEEE Access*, vol. 5, pp. 10 892–10 900, 2017.

[8] J. Del Ser, J. L. Lobo, E. Villar-Rodriguez, M. N. Bilbao, and C. Perfecto, "Community detection in graphs based on surprise maximization using firefly heuristics," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 2233–2239.

[9] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 554–560.

[10] G. Bello-Orgaz, H. D. Menéndez, and D. Camacho, "Adaptive k-means algorithm for overlapped graph clustering," *International journal of neural systems*, vol. 22, no. 05, p. 1250018, 2012.

[11] R. Francisquini, V. Rosset, and M. C. Nascimento, "Ga-lp: A genetic algorithm based on label propagation to detect communities in directed networks," *Expert Systems with Applications*, vol. 74, pp. 127–138, 2017.

[12] F. Folino and C. Pizzuti, "An evolutionary multiobjective approach for community discovery in dynamic networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1838–1852, 2014.

[13] S. B. Mathias, V. Rosset, and M. C. Nascimento, "Community detection by consensus genetic-based algorithm for directed networks," *Procedia Computer Science*, vol. 96, pp. 90–99, 2016.
[14] S. Li, Y. Chen, H. Du, and M. W. Feldman, "A genetic algorithm with local search strategy for improved detection of community structure," *Complexity*, vol. 15, no. 4, pp. 53–60, 2010.
[15] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.
[16] Y. Park and M. Song, "A genetic algorithm for clustering problems," in *Proceedings of the third annual conference on genetic programming*, vol. 1998, 1998, pp. 568–575.
[17] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.
[18] B. Klimt and Y. Yang, "Introducing the enron corpus." in *CEAS*, 2004.