Optimización de Ataques a Redes Complejas Mediante un Algoritmo de Colonias de Abejas Artificiales

Manuel Lozano[1], Carlos García-Martínez[2], Francisco J. Rodríguez[3], and Humberto M. Trujillo[4]

[1] Dpto. de Ciencias de la Computación e I. A., Universidad de Granada
lozano@decsai.ugr.es,
[2] Dpto. de Informática y Análisis Numérico, Universidad de Córdoba
cgarcia@uco.es
[3] Dpto. de Ciencias de la Computación, Universidad de Extremadura
fjrodriguez@unex.es
[4] Dpto. de Metodología de las Ciencias del Comportamiento, Universidad de Granada
humberto@ugr.es

Abstract. En los últimos años, ha crecido el interés en formular como un problema de optimización la tarea de concebir ataques efectivos que causen el máximo daño sobre redes complejas. En este caso, los ataques se modelan como un proceso de eliminación de k vértices del grafo que representa la red. En este trabajo, seguimos esta línea de investigación presentando un problema de optimización que concierne la selección de los nodos a eliminar con el objetivo de minimizar el máximo valor de intermediación en el grafo residual. La intermediación es una medida de centralidad bien conocida que evalúa la importancia de los nodos de la red de acuerdo a su participación en los caminos más cortos. La relevancia de este indicador dentro de la tecnología actual disponible para el análisis de redes nos ha llevado a plantear esta técnica para planificar ataques efectivos sobre redes.

Además, para abordar el problema de optimización, proponemos un algoritmo de colonias de abejas artificiales, que es una técnica de inteligencia colectiva inspirada en el comportamiento de las abejas cuando realizan la búsqueda de comida. Nuestra propuesta explota el conocimiento útil sobre el problema que se obtiene de la exploración de las fuentes de comida, aplicando una destrucción parcial de las soluciones escogidas y una reconstrucción heurística de las mismas. Mediante el análisis experimental de los resultados mostramos el buen comportamiento del algoritmo propuesto, con respecto a métodos de la literatura que pueden adoptarse para enfrentarse con el problema, tal como el método de ataque secuencial basado en centralidad.

Keywords: Algoritmo de colonias de abejas artificiales, Centralidad de intermediación, Detección de nodos críticos.

Actas de la XVII Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 103-112

1 Introducción

La teoría sobre las redes y sus aplicaciones están presentes en una amplia variedad de campos científicos (física, ingeniería, sociología, psicología, criminología, epidemiología y biología, entre otros), debido a la inherente habilidad de la redes para representar, de forma lógica, las importantes relaciones (aristas) entre los elementos estructurales de los sistemas complejos (nodos). La optimización de procedimientos para desmantelar eficientemente redes complejas está recibiendo mucha atención actualmente, desde un punto de vista práctico [5]. Por ejemplo, estos métodos pueden ayudar a las agencias de inteligencia a desarticular redes terroristas de corte yihadista [2] y también pueden facilitar el diseño de estrategias de vacunación capaces de contener la propagación de una epidemia pandémica [14]. En términos de un procedimiento de ataque, el reto consiste en encontrar un subconjunto de nodos o aristas cuya eliminación causaría un grado alto de daño sobre la red. Un camino eficaz, considerado con frecuencia para atacar un grafo consiste en borrar vértices teniendo en cuenta su importancia en el funcionamiento estructural de la red. En esta línea, los métodos tradiciones de ataque ordenan los nodos de forma decreciente en base a algún índice de centralidad; son los denominados *métodos de ataque basados en centralidad* [7].

Los índices de centralidad son métricas fundamentales para el análisis de redes. Miden cómo de central e importante es un vértice dentro de la red. Algunos de ellos, tal como la centralidad de grado, reflejan propiedades locales del grafo subyacente, mientras que otros, como la *centralidad de intermediación* (CI), aportan información sobre la estructura global de la red, ya que se basan en la computación y conteo de los caminos más cortos. Específicamente, CI cuantifica la importancia de un vértice basándose en su ocurrencia en los caminos más cortos entre todos los posibles pares de vértices del grafo [1]. Esta medida es útil para identificar los nodos que son más críticos con respecto a su importancia en transmitir información entre todos los pares de nodos en la red.

Una estrategia popular para producir ataques efectivos sobre redes se basa en el concepto de *nodos críticos*. Son nodos cuya eliminación causa la máxima fragmentación de la red, de acuerdo a alguna métrica predefinida [2,15]. El problema de optimización correspondiente se conoce como *problema de la detección de nodos críticos* (DNC). Aunque algunos autores han sugerido la conveniencia de formular instancias de DNC en base a la medida CI [3], pocas han sido las propuestas concretas que se han presentado en la literatura [6]. Esto puede deberse al hecho de que cualquier problema de optimización combinatoria que involucre a CI será extremadamente complicado, dado que esta medida es computacionalmente muy costosa [9].

En este trabajo presentamos un problema de optimización, denominado *Min-Max CI*, que es una instancia de DNC con el objetivo de minimizar el máximo valor de CI en el grafo obtenido tras la eliminación de k nodos. Con ello, se pretende degradar la estructura de la red de tal forma que los actores clave que surjan en el grafo residual tengan la menor influencia posible en el flujo de información. En otras palabras, la meta es evitar la aparición de líderes fuertes en

CI en el grafo resultante. Dos hechos determinantes nos han motivado a abordar este difícil y desafiante problema:

1. El reciente desarrollo del *algoritmo de colonias de abejas artificiales* (en inglés, artificial bee colony; ABC) [8], el cual está siendo empleado exitosamente para resolver un amplio espectro de problemas de optimización NP-duros [8,12].
2. La aparición de *algoritmos de actualización* de CI que son capaces de recalcular los valores de CI de todos los nodos de un grafo en respuesta a posibles modificaciones en su estructura (inserciones y borrados de nodos y aristas) de forma más rápida que si se calculan desde el principio [10].

Nuestro objetivo con este trabajo es desarrollar un algoritmo ABC competitivo para Min-Max CI. Su componente clave es una técnica de vecindario efectiva que se beneficia del uso de un algoritmo de actualización de CI, que permite acelerar la convergencia de ABC hacia zonas prometedoras dentro del espacio de búsqueda de este difícil problema.

El resto del artículo se estructura de la siguiente manera. La Sección 2 presenta Min-Max CI, una nueva instancia de DNC basada en CI que, hasta lo que conocemos, nunca ha sido definido explícitamente en la literatura. La Sección 3 detalla el algoritmo ABC propuesto para este problema. La Sección 4 expone los experimentos que analizan nuestro ABC en diferentes contextos. La Sección 5 finaliza con las conclusiones del trabajo.

2 El Problema Min-Max CI

Una red compleja puede formalizarse mediante un grafo simple no dirigido y sin pesos, $G(V, E)$, donde V representa el conjunto de nodos (vértices) ($|V| = n$) y E representa el conjunto de aristas ($|E| = m$). La medida CI fue propuesta por Anthonisse [1] para cuantificar la importancia de un vértice de acuerdo a la fracción del número de caminos más cortos que pasan por él. Formalmente, el valor de CI de un nodo v en G, $ci(G, v)$, se define como:

$$ci(G, v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

donde σ_{st} es el número total de caminos más cortos desde el vértice s al vértice t y $\sigma_{st}(v)$ es el número total de caminos más cortos desde el vértice s al vértice t que pasan por v. Ahora, definimos el CI global de G, $CI(G)$, como:

$$CI(G) = \max_{r \in V} ci(G, r).$$

Estamos empleando el operador *máximo* como un operador de agregación de información para obtener un único valor de CI, el cual proporciona información sobre la capacidad de acción de los actores clave en la red. Presentamos un problema de optimización, denominado Min-Max CI, que es una instancia de DNC

con una función objetivo que está basada en CI. Específicamente, el problema radica en descubrir un conjunto de k vértices de G tales que tras su eliminación dejarán al correspondiente grafo resultante con el menor valor de CI posible. Sea $S \subseteq V$ el subconjunto de los k nodos a eliminar de G y $G^S = G(V \setminus S, \{(i,j) \in E | i,j \in V \setminus S\})$ el grafo resultante tras el borrado de estos nodos. Min-Max CI consiste en minimizar el valor de $CI(G^S)$ sobre el conjunto de todos los posibles subconjuntos de V con cardinal k:

$$\textbf{(Min-Max CI)} \quad \min_{S \subseteq V, |S|=k} CI(G^S).$$

Las estrategias tradicionales para modelar ataques a redes, los *ataques basados en centralidad* [7], se centran en eliminar vértices en base a una medida de centralidad y siguen dos esquemas distintos:

- *Ataques dirigidos simultáneos.* Se calcula la medida de centralidad para todos los vértices de la red y se eliminan aquéllos que tienen los valores más altos, todos a la vez.
- *Ataques dirigidos secuenciales.* Se calcula la medida de centralidad para los vértices de la red y aquél con mayor valor se elimina, resultando una nueva red donde los valores de centralidad de sus nodos pueden ser diferentes a los calculados inicialmente. Así, se recalcula la medida de centralidad para los nodos en el nuevo grafo y, de nuevo, se elimina el nodo más central. El proceso se repite hasta borrar la fracción especificada de nodos.

Iyer y otros [7] investigaron el efecto de estos esquemas de ataque considerando distintas medidas de centralidad, tales como grado, intermediación, cercanía, y eigenvector, sobre un variado conjunto de modelos de redes. Interesantemente, el esquema secuencial basado en CI resultó la técnica más efectiva para degradar la estructura de las redes.

El ataque secuencial basado en CI (ASCI) puede emplearse como una heurística voraz simple para resolver Min-Max CI. Ya que, en cada paso, se borra el vértice con el valor más alto de CI, cabe esperar que aquellos vértices que permanecen en el grafo tengan valores de CI más bajos, y entre ellos, el que será el nuevo líder con mayor CI. Es por ello, que consideramos a ASCI como método de referencia en nuestro trabajo.

Min-Max CI es un caso específico de DNC. Aunque durante la última década han surgido diferentes instancias de DNC [15], el caso que mayor atención ha recibido es el definido por Arulselvan y otros [2], donde la red óptima es aquella con máxima fragmentación y que simultáneamente muestra la mínima varianza entre el número de vértices de sus componentes conectadas. En otras palabras, lo que se persigue es que la red resultante contenga un número alto de componentes, cada una conteniendo un número similar de vértices. Nos referiremos a este problema como DNC-A. Arulselvan y otros trataron este problema desde un punto de vista práctico, proponiendo un modelo de programación lineal entera y una aproximación heurística basada en un algoritmo voraz y una técnica de búsqueda local. Ciertamente, la naturaleza de NP-completitud de este problema [2] hace que las metaheurísticas representen la aproximación más adecuada para obtener soluciones cercanas a las óptimas en un tiempo razonable [11,14].

3 ABC para Min-Max CI

ABC es una metaheurística basada en poblaciones relativamente nueva que se inspira en el comportamiento de los enjambres de abejas cuando buscan comida. Consta de un conjunto de fuentes de comida y de tres clases de abejas (*obreras*, *supervisoras* y *scouts*). Las fuentes de comida representan soluciones al problema tratado. Su posición codifica la configuración de la solución mientras que la calidad está asociada con la cantidad de néctar que tiene. Las abejas simbolizan agentes que operan sobre las fuentes de comida para generar nuevas soluciones candidatas. Las obreras exploran el vecindario de la fuente que tienen asignada y comparten información con las supervisoras. Éstas tienden a seleccionar buenas fuentes de comida a partir de las cuales continuar con la exploración del vecindario. Las scouts se encargan de buscar nuevas fuentes candidatas de una forma más exploradora, normalmente considerando el espacio de soluciones completo del problema.

En las siguientes secciones describimos la forma en la que el esquema de ABC puede adaptarse para abordar Min-Max CI. La Sección 3.1 introduce el operador de vecindario. La Sección 3.2 presenta la visión general de la propuesta. Finalmente, la Sección 3.3 aporta los fundamentos sobre los que nos hemos basado para obtener una versión eficiente del algoritmo.

3.1 Operador de vecindario destructivo-constructivo

Partimos aclarando que una solución para Min-Max CI es un subconjunto de k vértices del grafo de entrada. Para explorar la vecindad de una solución, proponemos un operador de vecindario que está inspirado en el *algoritmo voraz iterativo*. En particular, elimina un número de componentes de la solución y después opera como ASCI para obtener una nueva solución completa. Nótese que la destrucción de una solución al quitar los nodos seleccionados supone reconstruir el grafo inicial con estos nodos y sus correspondientes arcos.

El número de elementos borrados por este operador se determina mediante un parámetro ω. A partir de una solución S, el operador procede escogiendo de forma aleatoria estos elementos, los elimina y, después, genera la nueva solución S' (el vecino) con los vértices restantes en S y con nuevos nodos que se añaden siguiendo la heurística voraz ASCI, hasta que su cardinal alcanza el valor k.

El grado de diversificación e intensificación introducido por el operador de vecindario propuesto puede ajustarse fácilmente variando el parámetro asociado ω. Altos valores para ω van a causar que nuestro operador tienda a la diversificación ya que S' podrá ser bastante diferente a S. Por otro lado, bajos valores para este parámetro implicarán que S y S' puedan compartir bastantes elementos en común, promoviendo la intensificación.

3.2 Esquema general del algoritmo ABC propuesto

El algoritmo de nuestra propuesta de ABC comienza generando una solución inicial factible mediante el algoritmo ASCI. Después, produce una población de

soluciones (fuentes de alimentos) mediante el operador de vecindario, actuando como un agente de diversificación (es decir, con un valor alto para ω, ω_{div}), sobre la mejor solución encontrada hasta el momento. A continuación, se repiten los siguientes pasos hasta que se cumple el criterio de parada establecido:

- *Fase de las abejas obreras*: cada obrera explora las proximidades de su fuente de comida mediante el operador de vecindario configurado para que actúe como agente de intensificación (es decir, con un valor bajo para ω, ω_{int}). La nueva solución se acepta si es mejor que la fuente de comida previa.
- *Fase de las abejas supervisoras*: estas abejas escogen una fuente de comida aplicando un torneo binario. A continuación, las supervisoras aplican los mismos pasos que las abejas obreras.
- *Fase de las abejas scout*: las fuentes de comida que no se han mejorado durante *Limite* iteraciones consecutivas son abandonadas y reemplazadas por una nueva solución generada por el operador de vecindario en su formato de diversificación.

El algoritmo ABC requiere que se establezcan dos parámetros de control: NP es el número de fuentes de comida a manejar, que coincide con el número de abejas obreras y supervisoras y *Limite* es el número de iteraciones sin que se produzca una mejora antes de abandonar una fuente de comida. Además, los valores para ω_{div} y ω_{int} deben elegirse de forma adecuada. Al finalizar, el algoritmo devuelve la mejor solución generada durante su ejecución.

3.3 Implementación eficiente

Dos procedimientos clave de ABC, el operador de vecindario y el procedimiento ASCI, incorporan las siguientes acciones sobre un grafo: (1) Calcular CI para todos los vértices del grafo, (2) Determinar el vértice con el valor más alto de CI y quitarlo del grafo, (3) Recalcular CI para todos los restantes vértices y (4) Repetir los pasos 2 y 3 hasta eliminar k vértices del grafo.

Para obtener CI, se necesitan todos los caminos más cortos en el grafo, lo que supone que el cálculo de esta medida es notoriamente costosa (el mejor algoritmo conocido, el algoritmo de Brandes [4], tiene una complejidad en tiempo de $O(nm)$ para grafos sin pesos). De esta forma, el proceso indicado anteriormente no es factible desde un punto de vista práctico, incluso para grafos pequeños. Así, para poder poner en marcha nuestro ABC, se hace necesario incorporar algoritmos que proporcionen un cálculo más rápido de CI.

Recientemente se han propuesto métodos para actualizar los valores de CI de los nodos de un grafo tras un cambio en su estructura que son más rápidos que al calcularlos de nuevo con el método de Brandes [10]. En base a la disponibilidad de estos métodos, hemos decidido implementar los pasos que están implicados en el procesamiento del operador de vecindario y en el algoritmo ASCI como sigue: (1) Invocar el algoritmo de Brandes para calcular CI para todos los vértices del grafo, (2) Determinar el vértice con el valor más alto de CI y quitarlo del grafo, (3) Aplicar el algoritmo de actualización de Lee y otros [10] para recalcular CI

para los restantes vértices y (4) Repetir los pasos 2 y 3 hasta eliminar k vértices del grafo.

Debemos destacar que la utilización del algoritmo de Lee y otros en el Paso 3 es especialmente importante para el diseño del operador de vecindario, ya que éste debe invocarse un número muy elevado de veces durante el procesamiento de ABC, por lo que debe ser lo más eficiente posible.

4 Experimentos

ABC ha sido implementado en C (las operaciones con grafos han sido realizadas usando *NetworKit* [13]) y el código fuente ha sido compilado con gcc 4.6. Los experimentos se han realizado sobre un ordenador con procesador Intel® Core™ i7 a 3,2 GHz y 24 GB de memoria RAM bajo Fedora™ Linux V15. Hemos considerado 6 modelos de redes bien conocidos para generar los grafos de prueba que se han usado para estudiar el comportamiento de ABC: Modelo *Erdös-Rényi* (ER), modelo *Clustered random graph* (CR), modelo *Random geometric graph* (RG), modelo *Barabási-Albert* (BA), modelo *Watts-Strogatz* (WS) y modelo *Forest fire* (FF). En concreto, se han creado 10 grafos de cada modelo con tamaños que aumentan de 100 a 1000 nodos.

De cara a encontrar la configuración más robusta del algoritmo ABC, se ha realizado un exhaustivo ajuste de parámetros. Los resultados nos han permitido llegar a la conclusión de que la configuración más competitiva emplea $NP = 20$, $Limite = 0.1n$, $\omega_{int} = 0.75|S|$ y $\omega_{int} = 0.5|S|$ (nótese que S es la solución alterada por el operador de vecindario y que $|S| = k$).

En las siguientes secciones, presentamos un análisis comparativo entre ABC y otros algoritmos que pueden aplicarse para resolver Min-Max CI, desde un punto de vista práctico. En particular, enfrentamos nuestra propuesta con ASCI (Sección 4.1) y con CNP1, un algoritmo que fue presentado para resolver DNC-A (Sección 4.2). El tiempo límite de ejecución para ABC y CNP1 es de $3.6n$ segundos.

Para comparar ABC con otro algoritmo X, hemos considerado dos medidas de eficacia, *%Éxito* (*%E*) y *%Mejora* (*%M*). *%E* representa el porcentaje de grafos de prueba para los cuales ABC encuentra mejores soluciones (estrictamente) que X. *%M* es la media en porcentaje de la desviación relativa entre la calidad de la solución encontrada por ABC y la obtenida por X ($\frac{X-ABC}{X}$) sobre aquellos grafos de prueba donde ABC es el ganador. Además, analizaremos estas medidas para los casos donde X es mejor que ABC.

4.1 Comparación de ABC con ASCI

En este trabajo, consideramos que ASCI es un método de referencia para abordar Min-Max CI, dada su relevancia en la literatura [15]. De hecho, este procedimiento se emplea dentro de ABC para obtener una de las soluciones iniciales. Así, debemos investigar detalladamente si nuestro ABC es capaz de mejorar las soluciones proporcionadas por este método constructivo. Para cada grupo de

grafos de prueba (según el modelo empleado para generarlos), la Tabla 1 muestra las medidas $\%E$ y $\%M$ al comparar ABC con ASCI, para $k = \{0.1n, 0.3n, 0.5n\}$. Nótese que, como ABC internamente invoca a ASCI, éste nunca va a ganarle. Las medidas de eficacia se han calculado también teniendo en cuentas todos los grafos testeados (véase última fila).

Table 1. ABC vs. ASCI

	$k = 0.1n$				$k = 0.3n$				$k = 0.5n$			
	ABC gana		*ASCI gana*		*ABC gana*		*ASCI gana*		*ABC gana*		*ASCI gana*	
Grafos	*%M*	*%E*	*%M*	*%E*	*%M*	*%E*	*%M*	*%E*	*%M*	*%E*	*%M*	*%E*
ER	2.3	70	-	0	12.7	100	-	0	11.3	90	-	0
CR	1	60	-	0	13	100	-	0	22.8	100	-	0
RG	22.7	100	-	0	47.2	90	-	0	53.3	90	-	0
FF	23.2	70	-	0	30.9	40	-	0	60.5	30	-	0
BA	6.2	30	-	0	40.3	100	-	0	62.5	20	-	0
WS	69.4	100	-	0	25.7	100	-	0	100	100	-	0
Todos	26.1	71.7	-	0	27.6	88.3	-	0	49.2	71.7	-	0

Podemos remarcar las siguientes observaciones sobre la Tabla 1. Si tenemos en cuenta todos los grafos probados, ABC fue capaz de devolver mejores soluciones que ASCI en más del 70% de los casos, con mejoras que superan el 25%, para todos los valores de k. En base a estos datos, podemos concluir que el modelo de metaheurística ABC nos ha permitido concebir un optimizador efectivo para Min-Max CI, el cual supera aceptablemente al algoritmo más popular en la literatura que puede emplearse para abordar este problema.

Los valores bajos de $\%M$ obtenidos para los grafos ER y CR, cuando $k = 0.1n$, se deben, probablemente, al hecho de que ASCI puede obtener soluciones bastante precisas para estos grafos, los cuales tienen topologías sin estructura. En estos casos, ABC pudo alcanzar mejores soluciones que ASCI (véase los valores relativamente altos para $\%E$), pero sin diferencias significativas en la función objetivo. Una situación similar ocurre para los grafos BA. ASCI pudo identificar con éxito los nodos que son responsables de conectar las diferentes partes de este tipo de grafos. Para $k = 0.1n$, la eliminación de estos nodos permite la generación de soluciones con alta calidad, las cuales no pueden ser mejoradas significativamente por ABC (véase los bajos valores para $\%E$ y $\%M$). Finalmente, los bajos valores para $\%E$ en los grafos FF ($k = 0.3n$ y $k = 0.5n$), y en los grafos BA ($k = 0.5n$), se debe al hecho de que ASCI pudo desmantelar casi completamente estas redes (logrando un valor de función objetivo de cero), lo cual es una situación inmejorable.

4.2 Comparación de ABC con una metaheurística para DNC-A

Como hemos visto en la Sección 2, DNC-A es el problema de optimización vinculado con la identificación de nodos críticos más estudiado en la literatura. Aunque este problema y Min-Max CI están relacionados, no se puede asegurar que un método desarrollado para el primero vaya a operar con precisión sobre el segundo. Sin embargo, podemos intuir que soluciones con alta calidad para DNC-A pueden ser prometedoras como soluciones para nuestro problema. La

minimización del número total de nodos conectados en el grafo promueve el decremento del valor de CI de muchos nodos, lo que afectará probablemente al valor global de CI. Así, debemos investigar el comportamiento de algoritmos para DNC-A como herramientas para resolver Min-Max CI. Concretamente, hemos implementado un algoritmo estado del arte para DNC-A, denominado CNP1, que fue propuesto por Pullan [11].

Hemos aplicado CNP1 sobre los 60 grafos de prueba usando el mismo límite de tiempo que ABC ($3.6n$ segundos). Las soluciones devueltas por CNP1 para DNC-A fueron evaluadas como soluciones para Min-Max CI. La Tabla 2 muestra la comparación entre estos resultados y los obtenidos por ABC.

Table 2. ABC vs. CNP1

	$k = 0.1n$				$k = 0.3n$				$k = 0.5n$			
	ABC gana		*CNP1 gana*		*ABC gana*		*CNP1 gana*		*ABC gana*		*CNP1 gana*	
Grafos	*%M*	*%E*	*%M*	*%E*	*%M*	*%E*	*%M*	*%E*	*%M*	*%E*	*%M*	*%E*
ER	27.2	90	4.1	10	45.2	90	0.3	10	48	100	-	0
CR	26.9	100	-	0	43.5	90	7.6	10	49.5	100	-	0
RG	58.7	30	38.4	70	81.8	100	-	0	96.9	100	-	0
FF	59.8	80	8.4	20	72	70	22.6	30	91.7	100	-	0
BA	71	100	-	0	89.3	100	-	0	100	100	-	0
WS	81	100	-	0	98.2	100	-	0	100	100	-	0
Todos	53.8	83.3	28.9	16.7	72.7	91.7	15.2	8.3	81	100	-	0

Los indicadores de efectividad globales mostrados en la Tabla 2 revelan que ABC es mejor que CNP1 (en términos de CI global). Este hecho es más patente para el caso de $k = 0.3n$ y $k = 0.5n$. Los componentes algorítmicos de ABC han sido diseñados para actuar adecuadamente frente a las características específicas de Min-Max CI, permitiendo que este algoritmo explote el conocimiento del problema mejor que un algoritmo competitivo para un problema similar. CNP1 intenta fragmentar el grafo en el mayor número de componentes (con tamaños similares) que es posible, sin tener en cuenta el valor de CI de los nodos en el grafo residual. Interesantemente, esta forma de proceder ha tenido una ventaja clara sólo para el caso de atacar los grafos RG con $k = 0.1n$.

5 Conclusiones

Hemos propuesto un algoritmo ABC para abordar una instancia de DNC cuyo objetivo consiste en minimizar el valor de CI global del grafo residual. Nuestro optimizador aplica un operador de vecindario destructivo-constructivo para generar nuevas soluciones candidatas cuando las abejas exploran el vecindario de las fuentes de comida. Una de sus características esenciales es que incorpora un algoritmo de actualización de CI que recalcula este índice de la forma más eficiente posible. El algoritmo propuesto ha sido estudiado de forma empírica sobre 60 grafos y se ha mostrado bastante competitivo con respecto a un algoritmo constructivo de referencia y frente a un algoritmo estado del arte para el caso de DNC que mayor relevancia ha tenido en la literatura. Por lo tanto, concluimos que esta metaheurística es una buena opción para este problema.

Pensamos que este trabajo es una contribución significativa ya que representa un punto de encuentro entre tres líneas de investigación actuales con gran trascendencia: DNC [11,15], CI [9,10] y ABC [8]. Para futuras investigaciones, pretendemos adaptar nuestra propuesta para su aplicación a redes grandes con miles de nodos. Específicamente, vamos a explorar la incorporación en ABC de algoritmos aproximativos para el cálculo de CI [9]. También proseguiremos con una línea abierta de aplicación de la herramienta presentada para desmantelar redes terroristas de corte yihadista.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad Español y por el Fondo Europeo de Desarrollo Regional (MINECO/FEDER) en el marco del Proyecto de Investigación con Referencia DER2015-63857-R y por el proyecto 18/16 CEMIX UGR-MADOC.

References

1. Anthonisse, J.M. The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam, 1971.
2. Arulselvan, A., Commander, C., Elefteriadou, L., Pardalos, P. Detecting critical nodes in sparse graphs. Comput. Oper. Res. 36 (2009) 2193–2200.
3. Borgatti, S.P. Identifying sets of key players in a social network. Comput. Math. Organiz. Theor. 12 (2006) 21–34.
4. Brandes, U. A faster algorithm for betweenness centrality. Journal of Mathematical Sociology 25(2) (2001) 163–177.
5. Deng, Y., Wu, J., Tan, Y.-J. Optimal attack strategy of complex networks based on tabu search. Physica A 442 (2016) 74–81.
6. Gunasekara, R.C., Mehrotra, K., Mohan, C.K. Multi-objective optimization to identify key players in large social networks. Soc. Netw. Anal. Min. 5 (1) (2015) 1-20.
7. Iyer S, Killingback T, Sundaram B, Wang Z. Attack robustness and centrality of complex networks. PLoS ONE 8(4): e59613 (2013).
8. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial Intelligence Review 42(1) (2014) 21–57.
9. Kourtellis, N., Morales, G.F., Bonchi, F. Scalable online betweenness centrality in evolving graphs. IEEE Trans. on Knowledge and Data Eng. 27(9) (2015) 2494–2506.
10. Lee, M.-J., Choi, S., Chung, C.-W. Efficient algorithms for updating betweenness centrality in fully dynamic graphs. Information Sciences 326 (2016) 278–296.
11. Pullan, W. Heuristic identification of critical nodes in sparse real-world graphs. Journal of Heuristics 21(5) (2015) 577–598.
12. Rodríguez, F.J., Lozano, M., García-Martínez, C., González-Barrera, J.D. An artificial bee colony algorithm for the maximally diverse grouping problem. Information Sciences 230 (2013) 183–196.
13. Staudt, C., Sazonovs, A., Meyerhenke, H. Networkit: an interactive tool suite for high performance network analysis. http://arxiv.org/abs/1403.3005 (2014).
14. Ventresca, M., Aleman, D. A derandomized approximation algorithm for the critical node detection problem. Computers and Operations Research 43 (2014) 261–270.
15. Veremyev, A., Prokopyev, O.A., Pasiliao, E.L. Critical nodes for distance-based connectivity and related problems in graphs. Networks, 66 (3) (2015) 170–195.