Detección de puntos débiles en redes utilizando GRASP y Path Relinking

Sergio Pérez-Peló
Dept. of Computer Sciences
Universidad Rey Juan Carlos
C/Tulipán, S/N, 28933, Móstoles, Spain
Email: sergio.perez.pelo@urjc.es

Jesús Sánchez-Oro
Dept. of Computer Sciences
Universidad Rey Juan Carlos
C/Tulipán, S/N, 28933, Móstoles, Spain
Email: jesus.sanchezoro@urjc.es

Abraham Duarte
Dept. of Computer Sciences
Universidad Rey Juan Carlos
C/Tulipán, S/N, 28933, Móstoles, Spain
Email: abraham.duarte@urjc.es

***Resumen*—El avance que se ha producido en la tecnología en los últimos años ha hecho que sea imprescindible disponer de herramientas que permitan monitorizar y controlar las debilidades que aparecen en redes. Debido al carácter dinámico de las redes, continuamente se conectan y desaparecen dispositivos de las mismas, siendo necesario reevaluar las debilidades que han aparecido tras cada modificación. La evaluación de estas debilidades tiene como objetivo proporcionar información de utilidad al responsable de la red para conocer qué nodos de la misma es necesario reforzar. Los continuos cambios crean la necesidad de disponer de herramientas que evalúen los puntos críticos en tiempos reducidos. En este trabajo se presenta un algoritmo metaheurístico basado en la metodología GRASP (*Greedy Randomized Adaptive Search Procedure*) para detectar puntos débiles en redes. Además, se propone un método de combinación basado en *Path Relinking* para mejorar la calidad de las soluciones generadas. Los resultados experimentales muestran la calidad de la propuesta comparándola con el mejor algoritmo encontrado en el estado del arte.**

***Index Terms*—α-separator problem, GRASP, Path Relinking, nodos críticos**

I. Introducción

En los últimos años la ciberseguridad se ha convertido en uno de los campos más relevantes para todo tipo de usuarios: desde empresas e instituciones hasta usuarios individuales. El incremento del número de ataques a diferentes redes, así como de la importancia de la privacidad en Internet, han creado la necesidad de tener redes más seguras, confiables y robustas. Un ciberataque a una compañía que conlleve pérdida de información personal de sus clientes puede resultar en un importante daño económico y social [1]. Además, los ataques de Denegación de Servicio (del inglés *Denial of Service, DoS*) y ataques de Denegación de Servicio distribuídos (del inglés *Distributed Denial of Service, DDoS*) han ido siendo cada vez más comunes, debido a que un ataque exitoso puede resultar en la deshabilitación de un servicio de un proveedor de Internet, por ejemplo. Además, si otros servicios dependen a su vez del servicio atacado, podría producirse un fallo en cascada, afectando a un mayor número de clientes [2].

Es importante identificar cuáles son los nodos más importantes en una red. Ésta es una cuestión de interés para ambas partes de un ciberataque: el atacante y el defensor. El primero estará interesado en deshabilitar dichos nodos con el objetivo de hacer la red más vulnerable, mientras que el segundo estará interesado en reforzar esos nodos importantes con medidas de seguridad más robustas. Además, ambas partes quieren consumir la mínima cantidad de recursos posibles para minimizar los costes derivados del ataque / defensa: por un lado, el atacante está interesado en ocasionar el máximo daño posible a la red empleando la mínima cantidad de recursos posible; por otro lado, el defensor querrá reforzar la red minimizando los costes de mantenimiento y seguridad. Por lo tanto, es interesante para ambas partes identificar cuáles son los puntos más débiles en la red.

Definimos una red como un grafo $G = (V, E)$, donde V es el conjunto de vértices ($|V| = n$) y E es el conjunto de aristas ($|E| = m$). Un vértice $v \in V$ representa un nodo de la red, mientras que una arista $(v_1, v_2) \in E$, con $v_1, v_2 \in V$ indica que hay una conexión en la red entre los vértices v_1 y v_2. Definimos un separador de una red como un conjunto de vértices $S \subseteq V$ cuya eliminación supone la separación de la red en dos o más componentes conexas. Formalmente,

$$V \setminus S = C_1 \cup C_2 \ldots \cup C_p$$
$$\forall\ (u, v) \in E^\star\ \exists\ C_i\ :\ u, v \in C_i$$

donde $E^\star = \{(u, v) \in E\ :\ u, v \notin S\}$.

Este trabajo se centra en encontrar un conjunto de vértices mínimo $S^\star$ que permita separar la red G en componentes conexas de tamaño menor que $\alpha \cdot n$. En términos matemáticos,

$$S^\star \leftarrow \arg\min_{S \in \mathbb{S}} |S| \quad : \quad \max_{C_i \in V \setminus S} |C_i| \leq \alpha \cdot n$$

Cabe mencionar que el número de componentes conexas resultantes no es relevante para este problema. La restricción actual del problema del separador α (α-SP) es que el número de vértices en cada componente conexa debe ser menor o igual que $\alpha \cdot n$. Este problema es $\mathcal{NP}$-completo para redes de topología general cuando se consideran $\alpha \geq \frac{2}{3}$ [3]. Se han propuesto algunos algoritmos en tiempo polinómico cuando la topología de la red es un árbol o un ciclo [4]. Sin embargo, estos algorimos requieren tener un conocimiento previo de la topología de la red, lo que no es usual en problemas de la vida real.

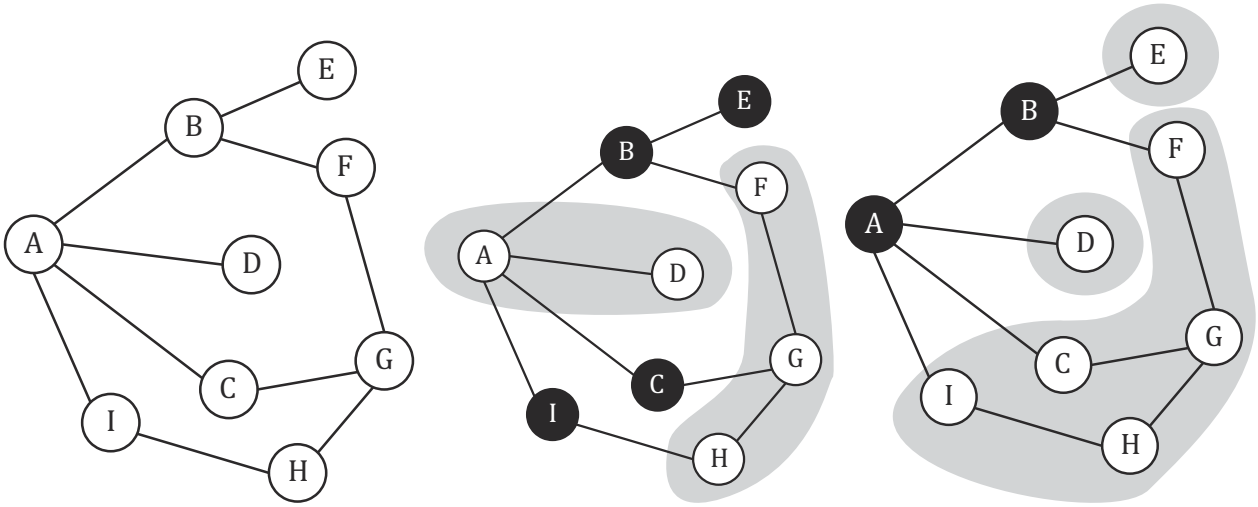

Figura 1: 1(a) Ejemplo de un grafo derivado de una red, 1(b) una solución factible con 4 nodos en el separador (`B,C,E`, y `I`), y 1(c) una solución mejor con 2 nodos en el separador (`A` y `B`)

Este problema ha sido abordado tanto desde la perspectiva exacta como la heurística. En concreto, se han presentado algoritmos que resuelven el problema en un tiempo polinómico para topologías como árboles o ciclos, así como un algoritmo voraz con una relación de aproximación de $\alpha \cdot n + 1$ [4]. Adicionalmente, un algoritmo heurístico para estudiar los separadores y los Sistemas Autónomos de Internet se propuso en [5]. Dependiendo del valor de α, el problema del separador α puede relacionarse con otros problemas bien conocidos. En particular, cuando $\alpha = \frac{1}{n}$, es equivalente al *minimum vertex cover problem*, y cuando $\alpha = \frac{2}{n}$ es análogo al *minimum dissociation set problem*. Por lo tanto, el problema del seaparador α es una generalización de estos problemas, que son también $\mathcal{NP}$-completos [6].

II. Propuesta algorítmica

Greedy Randomized Adaptive Search Procedure (GRASP), que en castellano se podría traducir como *procedimientos de búsqueda voraz, aleatorizados y adaptativos*, es un procedimiento multi-arranque que fue originalmente propuesto en [7] pero no se definió formalmente hasta [8]. Los procedimientos multi-arranque son un conjunto de procedimientos que intentan evitar quedar atrapados en óptimos locales reiniciando el procedimiento desde otro punto de inicio del espacio de búsqueda, lo que permite aumentar la diversificación de la estrategia de búsqueda. Esta metaheurística se ha utilizado recientemente con éxito en varios problemas de optimización [9]–[11].

Cada iteración en GRASP tiene dos fases bien diferenciadas: construcción y mejora. La primera tiene como objetivo generar una solución inicial de calidad para el problema, mientras que la segunda trata de mejorar la solución inicial a través de algún método de optimización local [12]. Tradicionalmente el optimizador de la segunda fase suele ser un procedimiento de búsqueda local sencillo, pero es común combinar esta metaheurística con procedimientos de búsqueda más complejos, como pueden ser Tabu Search [13] o Variable Neighborhood Search [14], entre otros, lo que prueba la versatilidad de la metodología GRASP.

II-A. Criterio voraz

La fase de construcción necesita definir un criterio voraz $g(v)$ que evalúe, para cada vértice $v \in V$, la relevancia de insertar dicho vértice en la solución. Para el problema que estamos tratando, α-SP, el criterio voraz debe indicar cuáles de los vértices que aún no han sido incluidos en la solución son nodos críticos en la red.

En este trabajo se propone la utilización de criterios utilizados en el área del análisis de redes sociales, de manera que se tratará de identificar qué vértices serían los más importantes de la red si ésta se tratara de una red social. En el análisis de redes sociales, un vértice es importante si mantiene a un gran número de usuarios de la red conectados entre sí [15], por lo que la analogía con el problema del α-SP es directa: un vértice es importante en la red si su desaparición hace que los dispositivos conectados a él dejen de estar conectados.

Existen numerosas métricas para evaluar la importancia de un usuario en una red social, siendo el cálculo de la mayoría de ellas muy costoso computacionalmente. Esto es debido a que varias de las métricas, como pueden ser *Betweenness* [16] o *PageRank* [17] necesitan conocer todos los caminos más cortos que pasan entre cada par de usuarios de la red. En este trabajo se utilizará la métrica denominada *closeness* [18], la cual considera que un vértice es importante si es alcanzable (i.e., se puede llegar a él a través de muchos caminos) por un elevado número de vértices de la red. Más formalmente, dada una red conexa G, el valor del *closeness* de un vértice se calcula como la suma de la longitud de las rutas más cortas entre el nodo en cuestión y el resto de nodos en el grafo.

Cabe destacar que la evaluación de esta métrica solo necesita conocer el camino más corto entre cada par de vértices, por lo que basta con realizar un recorrido en anchura (si la red no es ponderada) o aplicar el algoritmo de Dijkstra (si la red es ponderada). La idea que subyace bajo esta métrica es que, cuanto más central es un nodo, más cerca está de todos los demás nodos.

El valor del *closeness* de un nodo $v \in V$, definido como $C(v)$ se evalúa como el cálculo inverso de la suma de las distancias desde ese nodo v hasta el resto de nodos de la red $u \in V \setminus \{v\}$. En términos matemáticos,

$$C(v) \leftarrow \frac{1}{\sum_{u \in V \setminus S} d(v,u)}$$

donde $d(v,u)$ es la distancia entre los nodos v y u. En este trabajo, se considera que la distancia entre dos vértices $u, v \in V$ será la longitud del camino más corto que conecta u con v. Cabe destacar que esta métrica solo tiene sentido en componentes conexas, ya que de otra manera no sería posible encontrar un camino entre los vértices u y v.

II-B. Fase de construcción

El procedimiento de construcción parte de una solución vacía e iterativamente añade vértices a dicha solución hasta que ésta es factible. En el caso del α-SP, inicialmente ningún vértice ha sido añadido al separador, por lo que es necesario añadir vértices al mismo hasta cumplir la restricción de que la red se encuentra dividida en componentes conexas de tamaño inferior a $\alpha \cdot n$. El Algoritmo 1 muestra el pseudocódigo del método de generación de soluciones propuesto.

Algorithm 1 *Construir*(G, β)

1: $v \leftarrow Random(V)$
2: $S \leftarrow \{v\}$
3: $CL \leftarrow V \setminus \{v\}$
4: **while not** $esFactible(S, G)$ **do**
5: $\quad g_{\text{mín}} \leftarrow \text{mín}_{v \in CL}\, C(v)$
6: $\quad g_{\text{máx}} \leftarrow \text{máx}_{v \in CL}\, C(v)$
7: $\quad \mu \leftarrow g_{\text{máx}} - \beta \cdot (g_{\text{máx}} - g_{\text{mín}})$
8: $\quad RCL \leftarrow \{v \in CL \ : \ C(v) \geq \mu\}$
9: $\quad v \leftarrow Random(RCL)$
10: $\quad S \leftarrow S \cup \{v\}$
11: $\quad CL \leftarrow CL \setminus \{v\}$
12: **end while**
13: **return** S

El procedimiento comienza eligiendo al azar el primer vértice que va a ser incluido en la solución (pasos 1-2). La elección aleatoria de este primer elemento permite aumentar la diversidad de las soluciones generadas en la fase de construcción. A continuación, se crea la lista de candidatos *CL* con todos los vértices de la red salvo el elegido inicialmente (paso 3). El método de construcción entonces añade un nuevo vértice a la solución hasta que ésta satisfaga las restricciones del problema (pasos 4-12). La función *esFactible* tiene como objetivo comprobar que todas las componentes conexas de la red tienen un tamaño inferior a $\alpha \cdot n$ si se eliminan los nodos incluidos en la solución parcial S.

Tras ello, se calculan el mínimo y el máximo valor (pasos 5 y 6, respectivamente) para la función voraz elegida. En este caso, se utiliza el valor de *closeness* descrito en la Sección II-A. Estos dos valores se utilizan para calcular un umbral μ (paso 7) que limitará los vértices que entran a formar parte de la lista de candidatos restringida *RCL* (paso 8). El valor del umbral dependerá de un parámetro β (con $0 \leq \beta \leq 1$) que controla la aleatoriedad o voracidad del método. Por un lado, si $\beta = 0$, entonces $\mu = g_{\text{máx}}$ y, por tanto, solo se considerarán en la *RCL* aquellos vértices con el máximo valor de *closeness*, resultando en un método completamente voraz. Por otra parte, si $\beta = 1$, entonces $\mu = g_{\text{mín}}$, formando parte de la *RCL* todos los vértices de la *CL*, convirtiéndose así en un método totalmente aleatorio. De esta forma, cuanto mayor sea el valor del parámetro β, más aleatorio será el método, y viceversa. La Sección III discutirá cómo afecta el valor de este parámetro a la calidad de las soluciones obtenidas.

II-C. Fase de mejora

Las soluciones generadas en la fase de construcción se han construido mediante un procedimiento aleatorizado (a través del parámetro β), por lo que, en general, es posible aplicar un método de optimización local para mejorar la calidad de la solución. Aunque es posible utilizar cualquier tipo de optimizador en esta fase, el problema tratado requiere de la obtención de soluciones en períodos cortos de tiempo, por lo que se han descartado optimizadores complejos, optando por un procedimiento de búsqueda local tradicional.

Los procedimientos de búsqueda local tienen como objetivo encontrar el óptimo local de una solución con respecto a una vecindad predefinida. En primer lugar, es necesario definir qué vecindad va a explorar el procedimiento. En este trabajo se considerará la vecindad 2×1, la cuál contiene a todas las soluciones alcanzables desde una solución inicial S mediante la eliminación de dos vértices que estaban originalmente en la solución y la inserción de un nuevo vértice en la misma. Cabe destacar que cualquier movimiento dentro de esta vecindad que produzca una solución factible habrá mejorado el valor de la función objetivo, ya que habrá reducido el conjunto de vértices que necesitan ser eliminados de la red en una unidad. Más formalmente, la vecindad se define como:

$$N(v,S) \leftarrow \{S' \ : \ S' \leftarrow S \setminus \{u,v\} \cup \{w\}, \\ \forall\ u,v \in S \wedge\ \forall\ w \in V \setminus S\}$$

Existen dos métodos tradicionales para explorar la vecindad definida: *First Improvement* y *Best Improvement*. El primero recorre todas las soluciones de la vecindad, realizando siempre el primer movimiento que conduzca a una solución de mayor calidad, comenzando la búsqueda de nuevo desde la nueva solución. Sin embargo, *Best Improvement* primero explora todas las soluciones existentes en la vecindad, realizando el movimiento que lleve a la solución de mayor calidad.

En este trabajo se considerará únicamente la variante *First Improvement* debido a dos motivos principales. En primer lugar, *First Improvement* requiere de un menor coste computacional ya que no recorre la vecindad completa para realizar un movimiento y, como se ha mencionado en otras ocasiones, en este problema se requiere un tiempo de cómputo reducido. Por otra parte, dada la vecindad 2×1 definida, cualquier movimiento que lleve a una solución factible producirá una solución cuyo valor de la función objetivo se habrá mejorado en una unidad, sea cual sea el movimiento que se realice. Por tanto, para esta vecindad no es útil comprobar todos los posibles movimientos disponibles.

Teniendo en cuenta lo mencionado anteriormente, el procedimiento de búsqueda local propuesto para el problema del α-SP consiste en recorrer de manera aleatoria la vecindad de cada una de las soluciones generadas en la fase de construcción, parando cuando no se encuentre ninguna solución de mayor calidad en la misma, y devolviendo la mejor solución encontrada en la búsqueda.

II-D. Path Relinking

Path Relinking (PR) es un método de combinación de soluciones propuesto originalmente como una metodología para integrar estrategias de intensificación y diversificación en el contexto de la búsqueda tabú [19]. El comportamiento de PR se basa en explorar las trayectorias que conectan soluciones de alta calidad, con el objetivo de generar soluciones intermedias que, eventualmente, pueden ser mejores que las soluciones originales utilizadas para construir la trayectoria. Laguna y Martí [20] adaptaron *Path Relinking* en el contexto de GRASP como un método destinado a aumentar la intensificación. El algoritmo de *Path Relinking* opera sobre un conjunto de soluciones, llamado *Elite Set* (ES), que se encuentra ordenado siguiendo un criterio descendiente de acuerdo al valor de un criterio de calidad predefinido de cada solución. En este trabajo, consideramos como criterio de calidad el valor de la función objetivo de la solución. En concreto, el *Elite Set* se compone de las soluciones de mayor calidad generadas mediante el algoritmo GRASP. Este diseño generalmente se denomina estático [21], ya que en primer lugar se aplica GRASP para construir el conjunto de élite y posteriormente se aplica *Path Relinking* para explorar trayectorias entre todos los pares de soluciones en el ES. Por otra parte, el diseño dinámico considera la actualización del *Elite Set* cada vez que se explora una trayectoria.

Dadas una solución inicial, S_i y una solución guía S_g, *Path Relinking* crea una trayectoria de soluciones intermedias desde S_i hasta S_g a través de la inserción en S_i de características pertenecientes a S_g que aún no se encuentran en S_i. En el contexto de α-SP, se propone la eliminación de un vértice v que se encuentra en la solución que se está explorando en la trayectoria S' pero no está incluido en la solución guía, a la vez que se inserta un elemento u que no se encuentra en S' pero sí en S_g. Formalmente, se define entonces el movimiento *swap* de la siguiente manera:

$$Swap(S', v, u) \leftarrow S' \setminus \{v\} \cup \{u\} : \ v \in S' \setminus S_g, \ u \in S_g \setminus S'$$

En cada iteración, *Path Relinking* realiza un movimiento de *swap* que genera una nueva solución intermedia en la trayectoria. Cabe destacar que este movimiento produce, con alta probabilidad, una solución no factible, por lo que será necesario reparar la solución intermedia generada tras un movimiento $Swap(S', v, u)$. Para ello, tras el movimiento, se insertarán de manera aleatoria tantos nodos en la solución como sean necesarios para que la solución vuelva a ser factible.

Existen numerosas variantes de *Path Relinking* que difieren en cómo se construye la trayectoria. Tradicionalmente se consideran dos variantes de *Path Relinking*: *Greedy Path Relinking* (GPR) y *Random Path Relinking* (RPR). GPR, en cada iteración, genera una solución intermedia por cada elemento incluido en S_g que no está incluido en la solución intermedia S', continuando la trayectoria por la mejor solución intermedia de entre las generadas. Por otra parte, RPR selecciona aleatoriamente un elemento de entre los disponibles en S_g que no están incluidos en S' y genera una única solución intermedia con un movimiento *swap* donde se incluya dicho vértice. Mientras que GPR se centra en la intensificación, RPR tiene como objetivo la diversificación. De nuevo, la exploración exhaustiva de la trayectoria de GPR hace que no sea un método adecuado para el problema α-SP, debido al elevado coste computacional. En su lugar, se considera la variante RPR, menos exigente computacionalmente al explorar una única solución en cada etapa de la trayectoria. Para compensar la falta de intensificación en esta etapa, se aplicará el procedimiento de mejora descrito en la Sección II-C a la mejor solución encontrada en la trayectoria.

En este trabajo el *Elite Set* se compone de todas las soluciones generadas en la fase de construcción y mejora. Es importante aclarar que no se permite la entrada de soluciones ya existentes en el *Elite Set*, por lo que todas las soluciones combinadas serán siempre diferentes entre sí.

III. Resultados computacionales

Esta sección está dedicada a analizar el rendimiento de los algoritmos propuestos y comparar los resultados obtenidos con el mejor método previo encontrado en el estado del arte. Los algoritmos se han desarrollado en Java 9 y todos los experimentos se han llevado a cabo en un Intel Core 2 Duo 2,66 GHz con 4 GB de RAM.

El conjunto de instancias utilizadas en esta experimentación se ha generado utilizando el mismo generador de grafos propuesto en el mejor trabajo previo [22]. Específicamente, los grafos se generan utilizando el modelo Erdös Rényi [23], en el que cada nuevo nodo insertado tiene la misma probabilidad de estar conectado a un nodo ya existente en el grafo. Se ha generado un conjunto de 28 instancias de 100 vértices de diferente densidad (de 200 a 500 aristas).

La experimentación se encuentra dividida en dos partes diferentes: experimentación preliminar y experimentación final. La

primera está diseñada para ajustar el único parámetro del que requiere GRASP, β, que controla la aleatoriedad del método, mientras que la segunda se dedica a analizar el rendimiento de la mejor variante de GRASP junto con Path Relinking en comparación con el mejor algoritmo anterior encontrado en el estado del arte.

Todos los experimentos proporcionan las siguientes métricas: F.O., el valor de la función objetivo promedio; Tiempo (s), el tiempo promedio de cómputo medido en segundos; Desv.(%), la desviación porcentual promedio con respecto a la mejor solución encontrada en el experimento; y #Mejores, el número de veces que un algoritmo alcanza la mejor solución del experimento.

El experimento preliminar considerará un subconjunto de 20 instancias para evitar el sobreajuste del algoritmo. Para ajustar el valor del parámetro, se ha ejecutado la fase de construcción y mejora para generar un total de 100 soluciones por cada instancia, devolviendo la mejor solución encontrada, con $\beta = \{0{,}25, 0{,}50, 0{,}75, RND\}$, donde el valor *RND* indica que se obtiene un valor de β aleatorio en cada construcción. La Tabla I muestra los resultados obtenidos para cada valor.

Cuadro I: Rendimiento de GRASP (100 fases de construcción y mejora) con diferentes valores de β

β	F.O.	Tiempo (s)	Desv. (%)	#Mejores
0.25	29.75	18.37	2.31	11
0.50	29.90	18.92	2.91	12
0.75	31.65	18.81	7.09	7
RND	29.80	18.99	3.02	10

Como se puede observar, el valor de β que obtiene los mejores resultados es $\beta = 0{,}25$, que se corresponde con la variante que proporciona una menor aleatoriedad a la construcción. Este resultado parece indicar que el criterio de *closeness* utilizado es muy preciso para identificar los vértices críticos de la red. Por lo tanto, la diversificación se incluirá en mayor medida con la inclusión del método de combinación *Path Relinking*.

El experimento final está diseñado para comparar los resultados obtenidos incluyendo el método de combinación *Path Relinking* respecto al mejor algoritmo encontrado en el estado del arte, esta vez ejecutado sobre el total de instancias disponibles. El mejor algoritmo previo está basado en *Random Walks* combinado con *Markov Chains*. Debido a no haber recibido respuesta por parte de los autores previos, los resultados proporcionados son los obtenidos tras reimplementar detalladamente el método previo [24]. Los resultados se muestran en la tabla II.

Como se puede observar, el mejor algoritmo resulta ser *Path Relinking*, obteniendo el mejor resultado en todas las instancias del conjunto de datos utilizado. Comparando los resultados frente a GRASP, podemos concluir que *Path Relinking* es capaz de mejorar los resultados previos, quedando GRASP a una desviación del 7.42 %, a cambio de un mayor coste computacional (18.53 frente a 275.67 segundos). Aún con este incremento en el tiempo de cómputo, sigue requiriendo menos

Cuadro II: Comparativa de los resultados obtenidos por el algoritmo previo (RW) frente a los algoritmos GRASP y *Path Relinking* propuestos en este trabajo.

	F.O.	Tiempo (s)	Desv. (%)	#Mejores
GRASP	31.93	18.53	7.42 %	3
PR	29.43	275.67	0.00 %	28
RW	46.46	721.58	37.81 %	0

de la mitad del tiempo necesitado por el mejor método previo. Cabe destacar que, incluso GRASP de manera aislada, sin considerar *Path Relinking*, obtiene mejores resultados en todas las métricas que el método previo.

Para confirmar que existen diferencias estadísticamente significativas entre los métodos propuestos, se ha llevado a cabo el test no paramétrico de Wilcoxon. El p-valor menor que 0.00001 demuestra que sí que existen diferencias entre los métodos, resultando el siguiente ranking: PR (1.05), GRASP (1.95), y RW (3.00). Además, se ha llevado a cabo el test de Wilcoxon con signos para comparar algoritmos por pares. La comparativa de PR frente a RW ha resultado en un p-valor menor que 0.00001, al igual que el análisis entre GRASP y PR. Analizando estos resultados, podemos concluir que *Path Relinking* emerge como el mejor algoritmo del estado del arte para el problema del α-SP.

IV. Conclusiones

En este trabajo se ha presentado un algoritmo basado en GRASP para la detección de puntos críticos en redes. Además, las soluciones generadas se combinan utilizando *Path Relinking* para mejorar la calidad de las soluciones obtenidas. Los resultados obtenidos superan a las mejores soluciones encontradas en el estado del arte, todo ello respaldado por test estadísticos. El algoritmo GRASP es capaz de encontrar soluciones de calidad en cortos períodos de tiempo (menos de 20 segundos en promedio), mientras que *Path Relinking* es un procedimiento más costoso computacionalmente pero capaz de mejorar significativamente las soluciones encontradas por GRASP. Ambos métodos obtienen mejores soluciones que el método previo, basado en *Random Walks* combinado con modelos de Markov. La solución propuesta emerge como el mejor método en el estado del arte, pudiendo obtener soluciones de alta calidad en tiempo real con GRASP y mejorarlas posteriormente con *Path Relinking* en aquellas situaciones en las que el tiempo no sea crítico.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad, ref. TIN2015-65460-C2-2-P.

Referencias

[1] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez Gasca, R. Schulz, A. Stankovic, C. Taylor, and V. Vittal, "Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1922–1928, 2005.

[2] P. Crucitti, V. Latora, and M. Marchiori, "Model for cascading failures in complex networks," *Phys. Rev. E*, vol. 69, p. 045104, 2004.
[3] U. Feige and M. Mahdian, "Finding small balanced separators." in *STOC*, J. M. Kleinberg, Ed. ACM, 2006, pp. 375–384.
[4] M. Mohamed-Sidi, "K-Separator Problem. (Problème de k-Sèparateur)." Ph.D. dissertation, Telecom & Management SudParis, Èvry, Essonne, France, 2014.
[5] M. Wachs, C. Grothoff, and R. Thurimella, "Partitioning the Internet." in *CRiSIS*, F. Martinelli, J.-L. Lanet, W. M. Fitzgerald, and S. N. Foley, Eds. IEEE Computer Society, 2012, pp. 1–8.
[6] M. Garey and D. Johnson, *Computers and Intractability - A guide to the Theory of NP-Completeness*. San Fransisco: Freeman, 1979.
[7] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67 – 71, 1989.
[8] T. A. Feo, M. G. C. Resende, and S. H. Smith, "A greedy randomized adaptive search procedure for maximum independent set." *Operations Research*, vol. 42, no. 5, pp. 860–878, 1994.
[9] A. Duarte, J. Sánchez-Oro, M. G. C. Resende, F. Glover, and R. Martí, "Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization." *Inf. Sci.*, vol. 296, pp. 46–60, 2015.
[10] J. D. Quintana, J. Sánchez-Oro, and A. Duarte, "Efficient greedy randomized adaptive search procedure for the generalized regenerator location problem." *Int. J. Comput. Intell. Syst.*, vol. 9, no. 6, pp. 1016–1027, 2016.
[11] J. Sánchez-Oro, M. Laguna, R. Martí, and A. Duarte, "Scatter search for the bandpass problem," *Journal of Global Optimization*, vol. 66, no. 4, pp. 769–790, 2016.
[12] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2006, vol. 57.
[13] F. Glover and M. Laguna, "Tabu search," in *Handbook of combinatorial optimization*. Springer, 2013, pp. 3261–3362.
[14] P. Hansen and N. Mladenović, "Variable neighborhood search," in *Search methodologies*. Springer, 2014, pp. 313–337.
[15] J. Scott, *Social Network Analysis: A Handbook*. Sage Publications, 2000.
[16] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of mathematical sociology*, vol. 25, no. 2, pp. 163–177, 2001.
[17] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.
[18] M. E. Newman, "The mathematics of networks," *The new palgrave encyclopedia of economics*, vol. 2, no. 2008, pp. 1–12, 2008.
[19] F. Glover, *Tabu search and adaptive memory programming – advances, applications, and challenges*. Dordrecht, Massachusetts, USA: Kluwer Academic Publishers, 1996.
[20] M. Laguna and R. Martí, "Grasp and path relinking for 2-layer straight line crossing minimization." *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 44–52, 1999.
[21] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control and cybernetics*, vol. 29, no. 3, pp. 653–684, 2000.
[22] J. Lee, J. Kwak, H. W. Lee, and N. B. Shroff, "Finding minimum node separators: A markov chain monte carlo method," in *DRCN 2017 - Design of Reliable Communication Networks; 13th International Conference*, March 2017, pp. 1–8.
[23] P. Erdős and A. Rényi, "On random graphs," *Publicationes Mathematicae*, vol. 6, p. 290, 1959.
[24] B. Hassibi, M. Hansen, A. Dimakis, H. Alshamary, and W. Xu, "Optimized markov chain monte carlo for signal detection in mimo systems: An analysis of the stationary distribution and mixing time," *Signal Processing, IEEE Transactions on*, vol. 62, no. 17, pp. 4436–4450, 2014.