universidad de león

Escuela de Ingenierías Industrial, Informática y Aeroespacial

MÁSTER EN INGENIERÍA AERONÁUTICA

Trabajo de Fin de Máster

Análisis de robustez y eficiencia en la red europea de aeropuertos

Autor: **Javier J. Guerrero Rodríguez**

Tutor: **María Teresa Trobajo de las Matas**

Fecha: **(Julio, 2022)**

UNIVERSIDAD DE LEÓN

Escuela de Ingenierías Industrial, Informática y Aeroespacial

MÁSTER EN INGENIERÍA AERONÁUTICA

Trabajo de Fin de Máster

ALUMNO: Javier J. Guerrero Rodríguez

TUTOR: María Teresa Trobajo de las Matas

TÍTULO: Análisis de robustez y eficiencia en la red europea de aeropuertos

TITLE: Robustness and efficiency analysis of the European airport network

CONVOCATORIA: Julio, 2022

RESUMEN:

Las redes de transporte y, en particular, las redes de aeropuertos, son infraestructuras críticas, susceptibles de sufrir disrupciones en su funcionamiento de tipo aleatorio o de tipo malicioso. En este proyecto, por primera vez se analiza la robustez y eficiencia de la red europea de transporte aéreo frente a ataques dirigidos. A partir de una base de datos propia y actualizada (2019) con las rutas y pasajeros en los aeropuertos europeos, se realiza una caracterización topológica de la red como red compleja y se compara con otras redes similares. Se realizan simulaciones de ataques fuertes de nodo en base a cuatro métricas de centralidad de nodo diferentes: grado, intermediación, cercanía y autovalor y se cuantifica el daño a partir de métricas de robustez y eficiencia. Se detecta que la red colapsa a partir de un $30\,\%$ de nodos eliminados y se determina qué ataques son más dañinos sobre la red. Los resultados obtenidos dejan abiertas varias líneas de investigación, como la mejora en la cuantificación de la robustez y el daño o en el desarrollo de estrategias de mejora de las redes de aeropuertos que las haga más resistentes a ataques maliciosos.

ABSTRACT:

Transport networks, and airport networks in particular, are critical infrastructures, susceptible to random or malicious disruptions in their operation. In this project, for the first time, the robustness and efficiency of the European air transport network against targeted attacks is analyzed. Based on a proprietary and updated (2019) database with routes and passengers at European airports, a topological characterization of the network as a complex network is performed and compared with other similar networks. Simulations of strong node attacks are performed based on four different node centrality metrics: degree, betweenness, closeness, and eigenvalue and damage is quantified from robustness and efficiency metrics. It is detected that the network collapses from a $30\,\%$ of removed nodes and it is determined which attacks are more damaging on the network. The results obtained leave open several lines of research, such as improving the quantification of robustness and damage or developing strategies for improving airport networks to make them more resistant to malicious attacks.

Palabras clave: Red compleja; Red de aeropuertos; Estrategia de ataque; Métrica de robustez

Firma del alumno:	**VºBº Tutor:**

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El estudio de redes de transporte aéreo tiene una gran importancia hoy en día debido a que son infraestructuras críticas. Su importancia se ha visto acrecentada en las últimas décadas, convirtiéndose en uno de los medios de transporte más rápidos y seguros del mundo [6], sobre todo para conectar ciudades en distintos continentes o áreas aisladas como islas. De manera similar a otros modos de transporte, los aviones recorren rutas predefinidas para llegar de un punto a otro. El conjunto de aeropuertos caracteriza fundamentalmente la red de transporte aéreo, independientemente de las rutas seguidas por los aviones [7].

El conjunto de aeropuertos está formado por aeropuertos de tipo hub, aeropuertos medianos y aeropuertos pequeños. En ocasiones, debido a la estructura de la red, los aeropuertos medianos son estratégicos para la conectividad de la red. Identificar estos aeropuertos y aplicar medidas de prevención puede evitar perturbaciones graves sobre la red. Las disrupciones en los sistemas de transporte aéreo, ya sea debido a causas naturales o producidas por el ser humano, pueden causar grandes pérdidas socio-económicas. Por ejemplo, la erupción del volcán Eyjafjallajökull en 2010 tuvo un efecto devastador en la red de tráfico aéreo europeo, impidiendo los viajes aéreos en la mayor parte de Europa durante 6 días, causando pérdidas económicas a las aerolíneas de aproximadamente 1300 millones de euros y afectando a más de 10 millones de pasajeros [8]. La situación de pandemia que se produjo en 2020 y el estallido del conflicto bélico en Ucrania son otros dos ejemplos en los que se han producido disrupciones en el tráfico aéreo, bien por la brusca disminución del tráfico o bien por el cierre

de aeropuertos y, por tanto, la pérdida de conectividad aérea en el tráfico de personas y mercancías.

Para la realización de este proyecto se ha revisado literatura correspondiente a la caracterización topológica y al estudio de métricas de robustez en redes como la española [9], la estadounidense [10], la china [11] o la mundial [12], entre otras.

Este trabajo se inscribe en el marco de los estudios de Máster en Ingeniería Aeronáutica como punto final de una etapa. El interés constante por las redes de transporte aéreo y el tráfico de pasajeros durante los años de formación universitaria, así como por el procesamiento de datos, ha desembocado en este proyecto de fin de máster sobre el estudio de la red europea de aeropuertos (*European Air Transport Network EATN*). Dado que las redes de transporte presentan, en general, propiedades de las redes de escala libre [13], y que estas redes son razonablemente resistentes a ataques aleatorios en su funcionamiento, en este trabajo estamos interesados en analizar la topología de la red EATN y en el efecto que sobre la red pueden producir ataques intencionados o maliciosos.

Así pues, el objetivo general de este trabajo es analizar la robustez y eficiencia en redes complejas a través del caso de estudio de la red EATN. Como objetivos específicos, se han establecido los siguientes:

- Construir la base de datos de la red EATN actualizada
- Describir las propiedades topológicas de la red y compararla con las propiedades de otras redes de transporte aéreo
- Revisar el concepto de ataque en redes complejas en la literatura
- Proponer nuevos ataques dirigidos en redes complejas a partir de propiedades centrales de los nodos a priori y a posteriori en cada paso de los ataques
- Revisar las medidas de robustez y eficiencia frente a ataques maliciosos en redes complejas en la literatura
- Proponer nuevas medidas de robustez local y global en redes, así como medidas globales de la eficiencia de la red frente a ataques dirigidos

- Simular diferentes ataques sobre la EATN para determinar los más dañinos y analizar su robustez y eficiencia frente a disrupciones intencionadas en los aeropuertos

En cuanto a la estructura del presente documento, en primer lugar se presenta el estado del arte, donde se introduce por un lado la temática de estudio y la base teórica del mismo, es decir, las redes complejas, y por otro lado, se definen las métricas y ataques a utilizar y los daños producidos sobre la red.

A continuación, se expone la metodología utilizada en el proyecto, desde la obtención de la base de datos y limpieza de la misma hasta su procesamiento, que consiste en la simulación de los ataques realizados y su implementación. Además, se explica cómo se han calculado las métricas de robustez y eficiencia sobre las redes estudiadas.

Finalmente, en el último capítulo, se presentan los resultados, tanto las propiedades topológicas de la red como los resultados provenientes de las sucesiones de ataques realizados sobre la red. Tras la discusión de dichos resultados, se presenta una sección de conclusiones y otra en la que se proponen nuevas vías de estudio relacionadas.

Capítulo 2

Estado del arte

2.1. Redes de aeropuertos como redes complejas

La teoría de las redes complejas es una interpretación de la física estadística de la antigua teoría de los grafos, destinada a describir y comprender las estructuras creadas por las relaciones entre los elementos de un sistema complejo [14]. La teoría de redes complejas es de aplicación a muchos campos de ingeniería, matemáticas y ciencias sociales, ya que con ella se pueden estudiar redes complejas como redes informáticas e internet, redes tecnológicas, redes sociales, redes económicas, redes biológicas o sistemas de transporte.

Este estudio se centra en el análisis de sistemas de transporte como redes complejas. Las redes de transporte y, en concreto, las redes de transporte aéreo, pueden modelizarse dentro de la teoría de redes complejas. En una red de transporte aéreo, los aeropuertos son nodos, y las rutas entre aeropuertos son links o aristas del grafo.

En este documento, nos centraremos en redes de transporte aéreo, sus rutas regulares y el tráfico de pasajeros. Estas redes se caracterizan por ser de tipo *simple*, sin *loops* (conexiones entre un mismo nodo), y por que, a lo sumo, existe un enlace entre dos nodos. Consideraremos únicamente, además, redes *no dirigidas*, es decir, supondremos que si existe un vuelo directo desde el aeropuerto i con destino el aeropuerto j, existirá un vuelo con origen en j y destino en i con el mismo número de pasajeros al año.

Por otro lado, en ocasiones únicamente interesa conocer la existencia o no de un enlace (ruta directa) entre dos nodos, mientras que en otras es necesario aportar información adicional (número de vuelos, número de pasajeros, distancias geográficas...), conocida como *peso* sobre los enlaces. Es por ello que, en este trabajo, consideraremos redes *ponderadas* y *no ponderadas*:

- **Red ponderada:** también llamado grafo ponderado, está formada por un conjunto de nodos y un conjunto de enlaces pesados entre nodos. La ponderación de los enlaces puede ser información añadida sobre el tráfico de información o elementos o bien sobre los costes o distancias de los enlaces. Debido a que los grafos utilizados en este proyecto son no dirigidos, la ponderación del enlace es independiente de cuál sea el nodo de origen y de destino [12].
- **Red no ponderada:** su estructura es similar a la de una red ponderada, sin embargo, los enlaces no son ponderados, es decir, todos los enlaces tienen el mismo peso, que se suele establecer como 1. La única información relevante es si existe o no el enlace.

En este documento, las redes de aeropuertos se modelizan como grafos no ponderados $G = (U, E)$ y ponderados $G = (U, E, W(G))$, donde U es el conjunto de aeropuertos y $E \subset U \times U$ es el conjunto de links, enlaces o rutas directas entre los aeropuertos. Denotaremos $|U| = n$ y $|E| = m$. $W(G)$ es la matriz de pesos $n \times n$ cuyo elemento $w_{i,j}$ es el peso asignado al enlace (u_i, u_j). Si sólo interesa la estructura topológica del grafo, denotaremos $A(G) = (a_{ij})$ a la matriz de adyacencia binaria del grafo, cuyo elemento $a_{i,j} = 1$ si (u_i, u_j) es un enlace del grafo, 0 en caso contrario [9].

Un camino de longitud l entre dos nodos u_1 y u_{l+1} es una sucesión de l links de la forma

$$(u_1, u_2), (u_2, u_3), \ldots, (u_l, u_{l+1})$$

En este trabajo consideraremos redes inicialmente *conectadas* o *conexas*, es decir, en las que siempre existe al menos un camino entre dos nodos. Intuitivamente quiere decir que la red no puede separarse en diferentes grupos de nodos desconectados entre sí.

Finalmente, en la siguiente sección, utilizaremos el concepto de *geodésica* o *camino corto* entre dos nodos, es decir, que se corresponden con caminos de longitud mínima.

2.2. Métricas básicas de red

En este apartado, se describen algunas de las principales métricas de la red, que permiten describir propiedades topológicas o estructurales de la red.

2.2.1. Métricas de nodo

Las métricas de nodo tienen como objetivo medir la importancia de los nodos de la red. Dependiendo de lo que se considere más importante, se trabajará con unas u otras medidas.

- **Centralidad de grado (*Degree point centrality*):** el grado de un nodo u es la suma de todos los pesos de las aristas conectadas a cada nodo, es decir, es la suma total de adyacencias de un nodo u_k que corresponde a la suma de términos de la fila o columna k en la matriz de adyacencia $A(G)$. Un nodo con alta centralidad de grado tiene una alta capacidad de comunicación, debido a su conexión directa con un alto número de nodos de la red. Es una medida en cierto modo "local", ya que un nodo puede tener alta centralidad de grado, pero estar conectado a otros nodos que tengan pocas o nulas aristas, de forma que su conectividad en la red puede ser baja aunque su centralidad de grado sea alta [15]. En redes ponderadas, la métrica homóloga a la centralidad de grado es la centralidad de fuerza (*Strength centrality*), en la que se tienen en cuenta los pesos de las aristas o links y se utiliza la matriz de pesos $W(G)$.

- **Centralidad de intermediación (*Betweenness centrality*):** se calcula como:

$$B(u) = \sum_{s \neq t} \frac{\sigma_{st}(u)}{\sigma_{st}} \tag{2.1}$$

  donde $\sigma_{st}(u)$ denota el número de caminos más cortos que van del nodo s al nodo t y que pasan por el nodo u, y σ_{st} es el número de geodésicas que conectan u_s y u_t [16]. Un nodo cuyo valor de $B(u)$ sea alto es un nudo de comunicación de la red, de tal manera que tiene la capacidad de controlar una parte importante del flujo de información o del tráfico, debido a la alta proporción de nodos que se comunican a través de él [15].

- **Centralidad de cercanía (*Closeness centrality*):** se calcula de la forma:

$$C(u) = \frac{(n-1)}{\sum_{v \neq u} d_G(u,v)} \tag{2.2}$$

donde $d_G(u,v)$ indica la longitud del camino más corto entre el nodo u y el nodo v [17]. Esta métrica es el inverso de la distancia media desde un nodo inicial determinado a todos los demás nodos de la red [16]. Para calcular la centralidad de cercanía de grafos desconectados, en los que se tienen grupos de nodos conectados entre sí pero desconectados con el resto de grupos, es necesario recurrir a la fórmula de Wasserman y Faust, que proponen una fórmula mejorada para los grafos con más de un componente conectado. El resultado es "una relación entre la fracción de actores del grupo que son alcanzables, y la distancia media" de los actores alcanzables [18]. Es decir, se obtiene el valor de cercanía del nodo dentro del subgrafo conectado al que pertenece, ponderado por el tamaño de dicho subgrafo. Denotando n como el número de nodos en el grafo completo y n_u como el número de nodos en el subgrafo G_u conectado al que pertenece u se calcula como sigue:

$$C_{WF}(u) = \frac{n_u - 1}{n-1} \frac{n_u - 1}{\sum_{v \in G_u} d(u,v)} \tag{2.3}$$

- **Centralidad de autovector (*Eigenvector centrality*):** es el autovector del mayor valor propio de la matriz de adyacencia. Esta métrica mide la influencia de un nodo en la red, de modo que los nodos con una elevada centralidad de autovector también se conectan con otros nodos que tienen una elevada centralidad de autovector [12]. En redes ponderadas, la métrica es la misma teniendo en cuenta las ponderaciones de los nodos.

2.2.2. Métricas globales de la red

- **Densidad** d**:** describe la proporción de conexiones potenciales de una red que son conexiones reales. Una conexión potencial es una conexión que podría existir entre dos nodos, independientemente de si realmente existe o no.

- **Grado medio $\langle k \rangle$:** el grado medio de una red es el promedio de los grados de los nodos de la red. Denotaremos k_i como el grado del nodo i.

- **Diámetro D:** es la mayor distancia entre cualquier par de vértices [9]. Para hallar el diámetro de un grafo, primero hay que encontrar el camino más corto entre cada par de vértices. La mayor longitud de cualquiera de estos caminos es el diámetro del grafo. Denotaremos d_{ij} como el camino más corto entre los nodos i y j.

- **Longitud promedio L:** es el número medio de pasos a lo largo de los caminos más cortos para todos los pares posibles de nodos de la red. Es una medida de la eficacia del transporte de información o de masas en una red. Las redes con la propiedad de *mundo pequeño (small-world)* tienen un bajo valor de este coeficiente, similar al de la red aleatoria con el mismo número de nodos y grado medio [19].

- **Coeficiente de Clustering C:** es una medida del grado en que los nodos tienden a agruparse formando *cliques* [9]. Las redes de *mundo pequeño* tienen altos valores de este coeficiente, mayores que las redes aleatorias. Denotaremos C_i como el coeficiente de clustering del nodo i, de tal manera que [9]:

$$C_i = \sum \frac{a_{ij} a_{jk} a_{ki}}{k_i(k_i - 1)} \tag{2.4}$$

2.3. Ataques sobre la red

Existen varias formas de clasificar los ataques que se pueden realizar sobre una red.

- **Según el tipo de disrupción:** se distingue entre ataques aleatorios o dirigidos. Los ataques aleatorios son ataques no controlados y pueden estar asociados a fallos en los sistemas o equipos, incendios, fenómenos meteorológicos... Otro tipo son los ataques dirigidos, que suponen una amenaza sobre la red. Por ejemplo, grupos terroristas, grupos cibercriminales u otros grupos que, con ánimo de dañar la red, atacan los nodos más importantes.

- **Según el objeto atacado:** se consideran dos opciones, ataques de nodos o de aristas. En un ataque de nodos, cuando se elimina un nodo, se eliminan todas las aristas conectadas a dicho nodo, a diferencia de los ataques de aristas, en los que se eliminan links o aristas concretas de la red.

- **Según el número de objetos atacados:** se distinguen entre ataques completos o incompletos. Los completos son aquellos que eliminan todos los nodos o links de la red de forma sucesiva, y los incompletos no llegan a eliminar todos los elementos de la red.

A continuación se profundiza en los ataques de nodo por su relevancia para el presente documento.

Un ataque de nodo en una red compleja $G = (U, E)$, es una secuencia de nodos $A = (u_1, u_2, \ldots, u_s)$ con $s \leq n$, de tal manera que $u_i \in U$ y para cada i, j se tiene que $u_i \neq u_j$ [12].

Se puede utilizar la información topológica de los nodos de la red para generar ataques dirigidos sobre los mismos. Esto es, ordenar los nodos en función de una métrica elegida y eliminar de forma sucesiva los mismos. De esta manera, se distinguen:

- **Ataques fuertes o débiles:** en los ataques fuertes se elimina secuencialmente el nodo más importante en función de la métrica elegida para el ataque, mientras que en los ataques débiles, se elimina secuencialmente, en primer lugar, el nodo más débil.

- **Ataques no-interactivos e interactivos:** se tienen dos formatos para realizar una sucesión de ataques sobre una red. El primero se trata de ataques no-interactivos o estáticos (N), en los que se calcula inicialmente la secuencia de nodos de ataque y se eliminan los nodos secuencialmente hasta destruir toda la red. El segundo formato son los ataques interactivos (I), en los que, cada vez que se elimina un nodo se recalcula de nuevo la métrica del ataque para generar la nueva secuencia con los nodos restantes [20].

2.4. Daños a la red

Cuando se realizan ataques aleatorios o dirigidos sobre una red, ésta queda dañada perdiendo conectividad y capacidad de flujo o tráfico de información.

Uno de los objetivos de este trabajo es cuantificar el daño producido en la red cuando se suprimen nodos en ataques dirigidos. Los ataques producen, por una parte, cambios en la conectividad, especialmente cuando disminuye el número de nodos que se encuentran conectados. Se obtendrá un coeficiente que permita recoger esta información. Además, tal y como se ha mencionado, cuando se elimina un nodo, se eliminan todos sus links, lo que modifica la capacidad de la red para el tráfico de información u objetos. Se propondrá otro coeficiente para cuantificar este daño. Estos coeficientes serán útiles para comparar los ataques realizados sobre la red y determinar cuál es el más dañino.

Por otro lado, el cálculo de los coeficientes mencionados, puede ser de utilidad para determinar cuál de ellas es más resistente a los ataques en cuanto a su conectividad y tráfico. Con ello, se podría comparar la robustez de diferentes redes respecto de un determinado ataque. Continuando con esta idea, revisamos a continuación algunos modelos de redes clásicas en la literatura:

- **Grafos completos (K_n)**: un grafo completo es un grafo en el que cada par de vértices del grafo está conectado por una arista. El grafo completo con n vértices se denota K_n y tiene $\binom{n}{2} = n \cdot (n-1)/2$ aristas [1]. Los grafos completos son los más robustos frente a atques dirigidos o aleatorios, dado que la eliminación de un nodo en un grafo completo genera de nuevo otro grafo completo con un nodo menos que el anterior. Este tipo de grafos se puede observar en la figura 2.1.

- **Grafos en árbol:** Los árboles son grafos simples, conexos, que no contienen ningún ciclo. Representan una estructura jerárquica en forma de árbol. Los árboles pertenecen a la clase más sencilla de los grafos [2].

  Un caso particular de los grafos en árbol son los grafos en estrella. Un grafo en estrella es un árbol de n nodos con un nodo central conectado con todos los demás. El grafo en estrella de n nodos se denota S_n. Un ejemplo de este tipo de grafo se puede observar en la parte izquierda de la figura 2.2. Se trata del tipo de grafo menos robusto frente a ataques dirigidos de nodo, dado que siempre eliminan, en primer lugar, su nodo central, aislando en un solo paso al resto de nodos de la red.

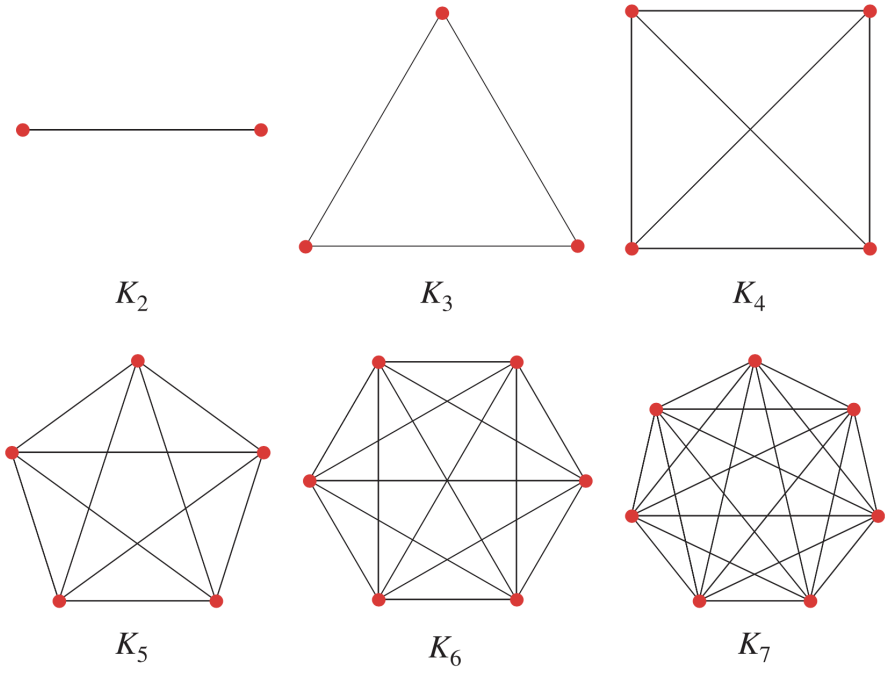


Figura 2.1: Esquema de distintos grafos completos K_n (Fuente: *Wolfram Research* [1])

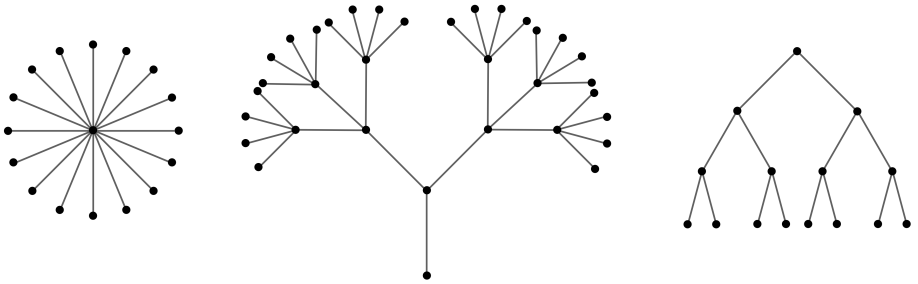


Figura 2.2: Esquema de distintos grafos en árbol. La figura de la izquierda se corresponde con la estrella S_{17} (Fuente: *P. Johnson* [2])

Capítulo 3

Metodología

En este capítulo se describe el proceso mediante el cual se ha generado y procesado la base de datos de la red europea de transporte aéreo (*European Air Transport Network EATN*). En la subsección 3.1.1 se describe la obtención de la base de datos y la organización de los mismos para su posterior procesamiento, en la subsección 3.1.2 se describe el proceso de construcción de la matriz de adyacencia binaria $A(G)$ y la matriz de pesos $W(G)$ y en la sección 3.2 se detalla la implementación de los ataques sobre la red.

3.1. Construcción de la red europea de transporte aéreo

3.1.1. Fuente de la base de datos

Eurostat es la oficina estadística de la Unión Europea. Se encargan, entre otras cosas, de desarrollar definiciones, clasificaciones y metodologías armonizadas para la elaboración de estadísticas oficiales europeas. Además, calculan datos agregados para la Unión Europea y zona euro y ponen las estadísticas a disposición del público general a través de su página web [4].

Debido a que en este proyecto es necesaria la construcción de una base de datos que contenga la red europea de aeropuertos junto con las rutas, se ha utilizado una base de datos de Eurostat. En concreto, la base de datos utilizada proviene de la información deta-

llada del transporte aéreo de pasajeros por país y sus rutas. Dentro de Eurostat, el ámbito del transporte aéreo contiene datos nacionales e internacionales intra y extracomunitarios. Proporciona datos de transporte aéreo de pasajeros (en número de pasajeros) y de carga y correo (en 1000 *toneladas*), así como datos de tráfico aéreo por aeropuertos, compañías aéreas y aviones.

Esta base de datos se subdivide en los datos de 35 países que reportan datos a Eurostat entre los que se incluyen los 27 Estados miembros de la UE, los países de la AELC y algunos otros países declarantes. Cabe mencionar que en este trabajo no se han utilizado los datos de Serbia, debido a que la estructura de datos del archivo es diferente a la del resto de países y esto presentaba problemas a la hora de importar los datos. Es preciso destacar que, en términos del número de aeropuertos, la información descartada correspondiente a Serbia supone un 0,56 % del total de aeropuertos de la red, por lo que no se considera crítico para los resultados.

Los datos se recopilan con arreglo a lo dispuesto en el Reglamento (CE) nº 1358/2003, por el que se aplica el Reglamento nº 437/2003 del Parlamento Europeo y del Consejo relativo a las estadísticas de transporte aéreo de pasajeros, carga y correo. Los datos del transporte aéreo se recogen a nivel de aeropuerto. Se utilizará la base de datos correspondiente al transporte aéreo de pasajeros. En cuanto a los aeropuertos incluidos en la base de datos, los aeropuertos que gestionan menos de 15000 pasajeros al año están excluidos del ámbito de aplicación del Reglamento.

La base de datos está estructurada de tal manera que, para cada país, incluye las rutas de los aeropuertos de dicho país con otros aeropuertos. Se ha comenzando descartando de los datos las rutas que tienen un origen con alguno de los 34 países de la base de datos y destino fuera de dichos países.

En cuanto al histórico de los datos, éstos se comenzaron a reportar en Eurostat en 2002, sin embargo, se toma como año de referencia el 2003. La periodicidad de los datos es mensual, trimestral y anual. Cabe destacar que para este proyecto se han utilizado los datos con pediodicidad anual correspondientes al año 2019. Se ha utilizado el año 2019 debido a que los datos correspondientes a los años 2020 y 2021 se encuentran afectados por la pandemia

mundial de coronavirus que es una pandemia actualmente en curso derivada de la enfermedad causada por el virus SARS-CoV-2. En los años 2020 y 2021, el transporte aéreo de pasajeros sufrió un impacto significativo causado por una gran reducción en la demanda debido a las restricciones de viaje, que obligó a las aerolíneas a cancelar vuelos y rutas.

Por último, la base de datos ofrece diferentes variables, entre las que se encuentran: pasajeros a bordo, pasajeros a bordo (llegadas), pasajeros a bordo (salidas), pasajeros transportados, asientos disponibles, vuelos comerciales de pasajeros, vuelos comerciales de pasajeros (llegadas), vuelos comerciales de pasajeros (salidas). Las unidades de medida de las variables mencionadas son pasajeros, asientos y vuelos, respectivamente. Para construir la red con pesos se ha utilizado la variable de pasajeros a bordo de las aeronaves, medida en número de pasajeros.

3.1.2. Construcción de las matrices de adyacencia y pesos

Tal y como se describe en la subsección 3.1.1, los datos están divididos por países y contienen una primera columna con el aeropuerto de origen y de destino, y el resto de columnas que contiene los datos de la métrica correspondiente en el periodo temporal elegido.

Se importan los archivos correspondientes a los 34 países en el editor de código *Visual Studio Code*, donde crearemos un script utilizando el lenguaje *Pyhton* para la limpieza de los datos. Inicialmente, se realiza un filtrado de los datos para utilizar únicamente la métrica de pasajeros a bordo, medida en número de pasajeros. Además, se guarda únicamente la columna correspondiente a los datos anuales de 2019.

Toda vez se tiene una estructura de datos bidimensional, con las rutas y los datos de pasajeros para el año 2019, se procede a separar la columna que contiene la ruta, para a continuación tener dos columnas, una con el aeropuerto de origen y otra con el de destino.

A continuación, se filtra y se eliminan aquellas rutas cuyo origen y destino sea el mismo. Por otra parte, se identifican las rutas dobles, es decir, aquellas para las que contemos con datos para la ida y la vuelta. En estos casos, la operación realizada ha consistido en calcular el promedio de los datos a la ida y a la vuelta. Esto se realiza porque el grafo objetivo es

un grafo no dirigido, en el que las aristas representan relaciones simétricas y no tienen un sentido definido, a diferencia de los grafos dirigidos.

Como ya se mencionó en la subsección 3.1.1, existen aeropuertos destino que no están en los 34 países de la base de datos. Por tanto, será necesario eliminar estos aeropuertos para que la red tenga congruencia.

Posteriormente, la operación realizada consiste en crear un vector con los aeropuertos únicos. Se construye una matriz cuadrada de tal manera que las filas y las columnas estén asociadas a los aeropuertos y se comienza a llenar de la siguiente manera:

- Para la construcción de la matriz con pesos, se busca en los vectores de la ruta el dato de los pasajeros para cada elemento de la matriz, y donde no existan datos se completará con NaN (*Not a Number*) en lugar de 0 para que no influyan a la hora de realizar cálculos.

- Para la construcción de la matriz sin pesos, se realiza una operación muy similar a la anterior, con la única diferencia de que en lugar de completar con el dato de pasajeros para cada elemento, se utiliza un 1. Donde no existan datos se completará con NaN de nuevo.

Debido a la naturaleza de la limpieza de datos realizada, se han eliminado todas las rutas cuyo origen y destino sea el mismo, por lo que la diagonal de ambas matrices estará compuesta de $NaNs$.

La última operación realizada antes de proceder a exportar las matrices es calcular la suma de cada fila para eliminar aquellas cuya suma sea cero. Se trata de aeropuertos que quizás hayan tenido rutas anteriormente, pero en 2019 no tuvieron ninguna ruta con los aeropuertos de la red.

Por último, se exportan las matrices en formato *csv*.

3.2. Simulación de ataques de nodo

3.2.1. Ataques realizados

En este documento se han realizado ataques completos, dirigidos, de nodo, no-interactivos e interactivos, fuertes y basados en métricas de centralidad de los nodos. Las métricas elegidas para definir las estrategias de los ataques se describen a continuación en la tabla 3.1.

Tabla 3.1: Resumen de los ataques realizados (Fuente: Elaboración propia)

Ataque	Descripción
DN	Centralidad de grado/fuerza no-interactivo
DI	Centralidad de grado/fuerza interactivo
BN	Centralidad de intermediación no-interactivo
BI	Centralidad de intermediación interactivo
CN	Centralidad de cercanía no-interactivo
CI	Centralidad de cercanía interactivo
EN	Centralidad de autovector no-interactivo
EI	Centralidad de autovector interactivo

3.2.2. Implementación de los ataques

Para realizar los ataques sobre la red, se ha creado un nuevo script en *Visual Studio Code* y se ha utilizado *Python* de nuevo para la realización de los cálculos. En la sección 2.3 se han definido los ataques que se han realizado sobre la red. Recordemos que se trata de ataques dirigidos, de nodo, y que se realizan ambas modalidades de ataque: no-interactivos (N) e interactivos (I).

Antes de nada, se importan las matrices de las redes con y sin pesos creadas anteriormente

en formato *csv*. La estructura del script utilizado para realizar los cálculos es muy modular, de forma que se han definido distintas funciones que son utilizadas sucesivamente a lo largo del script. Una de las funciones más importantes es la encargada de, a partir de una matriz y el ataque a realizar, obtener la secuencia de nodos A correspondiente al ataque.

Para calcular las secuencias de nodos derivadas de aplicar los 4 ataques mencionados, se utiliza el paquete *Networkx* que contiene funciones específicas para estudiar redes complejas. Este paquete permite construir un grafo no dirigido y, utilizando funciones del propio paquete, calcular los ataques elegidos en este estudio:

- **Centralidad de grado o fuerza (*Degree point centrality or Strength cetrality*):** se utiliza: *G.degree(airports[i], weight=w)*. Este comando se encuentra en un bucle que recorre todos los nodos (*airports*) del grafo. En el propio comando se indica si el grafo es con o sin pesos. El resultado es una matriz con el listado de aeropuertos y su grado, utilizando el algoritmo *mergesort* para ordenarlos de mayor a menor.

- **Centralidad de intermediación (*Betweenness centrality*):** el comando utilizado para este ataque ha sido: *nx.betweenness_centrality(G, k=None, normalized=True, weight=w, endpoints=True, seed=None)*, donde, a través de las diferentes opciones podemos indicar si el grafo es con o sin pesos. Aunque en este caso, los pesos deberán ser distancias, no pasajeros, por ello no podremos utilizar este comando para la red ponderada. También se puede indicar si se utilizan muestras para determinar la centralidad, si el cálculo es normalizado, si se incluyen los puntos finales y si se quiere incluir una semilla inicial. Para determinar los valores de estas variables se han seguido las recomendaciones de la documentación del paquete [21].

- **Centralidad de cercanía (*Closeness centrality*):** para el cálculo de este ataque se ha utilizado el comando: *nx.closeness_centrality(G, u=None, distance=None, wf_improved=True)*. Donde se puede calcular el ataque para un único nodo u, aunque en este caso será para todos. También se puede indicar si se quiere escalar por la fracción de nodos alcanzables, esto da la fórmula mejorada de Wasserman y Faust [21].

- **Centralidad de autovector (*Eigenvector centrality*):** el comando utilizado para calcular este ataque ha sido *nx.eigenvector_centrality(G, max_iter=100000, tol=1e-03, nstart=None, weight=w)*, donde se puede indicar el número máximo de iteraciones y la tolerancia. Cabe mencionar que se ha tenido que utilizar un número relativamente alto de iteraciones con una tolerancia media para que el ataque pudiera converger en todos los casos que se presentan en este proyecto. Se puede indicar un autovector inicial y si se quiere utilizar o no pesos.

Es preciso mencionar que, los ataques D y E son ataques que utilizan los pesos de tal manera que a mayor peso, mayor importancia del nodo. Sin embargo, ataques como B y C funcionan a la inversa, cuanto mayor peso, menor importancia del nodo. Esto conlleva que los ataques D y E se pueden utilizar con redes ponderadas en los que los pesos sean número de pasajeros, de vuelos u otras medidas de ganancia, mientras que B y C se utilizarán con distancias geográficas, ortodrómicas u otras medidas de coste. Por ello, en este estudio, para la red ponderada, sólo se aplicarán los ataques D y E [22].

Dependiendo de la modalidad del ataque realizado, N o I, se accede a la función únicamente al inicio de la sucesión de ataques para calcular la secuencia de nodos A en un ataque del tipo N, y se accede en cada paso de la sucesión de ataques en un ataque I.

Para ordenar los nodos tras calcular el valor de la métrica de nodo seleccionada, se ha elegido utilizar un algoritmo de ordenación determinista y estable, es decir, que el resultado tras la ordenación no sea aleatorio y que en caso de tener elementos iguales, los ordena en el mismo orden en el que se encontraban en la lista no ordenada. El algoritmo elegido es *mergesort*, que divide continuamente una lista en múltiples sublistas hasta que cada una tiene un solo elemento, y luego reordena esas sublistas en una lista ordenada. El esquema de funcionamiento de este algoritmo se puede observar en la figura 3.1. Cabe mencionar que en caso de empate, no se utiliza ninguna forma de deshacer los empates que utilice información adicional sobre la red.

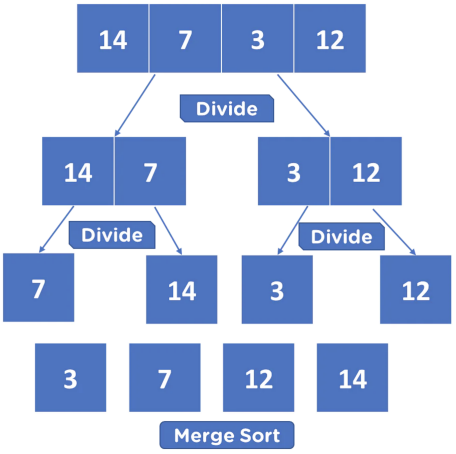


Figura 3.1: Esquema de funcionamiento del algoritmo de ordenación *mergesort* (Fuente: *SimpliLearn Solutions* [3])

3.3. Métricas

En cada paso de una sucesión de ataques se crea una red nueva. Esto abre la posibilidad de calcular métricas locales, es decir, métricas calculadas para cada red que se va creando a cada paso, y métricas globales, con las que podremos medir la robustez de la red ante un ataque determinado después de la eliminación de todos sus nodos. Estas métricas servirán para cuantificar el daño ocasionado a la red. A continuación, se describen alguna métricas locales y globales relevantes para este estudio.

3.3.1. Métricas locales de robustez y eficiencia

Son calculadas para la red inicial con todos los nodos y de nuevo cada vez que se elimina un nodo. Las métricas locales a describir serán el tamaño de la componente gigante de la red, las aristas o links que sobreviven en cada paso y la eficiencia.

- **Tamaño de la componente gigante *(GCC, Giant Connected Component)*:** dado que los ataques que se describen en la sección 3.2 atacarán la red hasta desintegrarla por completo (eliminar el último nodo), el tamaño del mayor subgrafo conectado es una métrica interesante para medir la funcionalidad de las redes [23]. A lo largo del presente documento, se denota $GCC(k)$ a la componente gigante en el paso k, tras la

eliminación de los primeros k nodos. Para la presentación de resultados, en lugar de utilizar los valores absolutos de los tamaños de las componentes gigantes de cada red, utilizaremos el valor relativo de los tamaños, es decir:

$$R(k) = \frac{|GCC(k)|}{|GCC(0)|} \tag{3.1}$$

- **Links que sobreviven *(SL, Survived Links)*:** en cada paso de la sucesión de ataques se elimina un nodo. Esto conlleva a que se eliminan todas las aristas o links conectados con dicho nodo. Se considera que la métrica que expresa los links que sobreviven en cada paso de la sucesión de ataques es interesante para estudiar la funcionalidad y robustez de la red. A lo largo del documento, se expresará la medida de links que sobreviven como $SL(k)$ en el paso k. Al igual que con anterioridad, se utiliza el valor relativo:

$$R^*(k) = \frac{SL(k)}{SL(0)} \tag{3.2}$$

- **Eficiencia *(E, Efficiency)*:** la eficiencia de un par de nodos en un grafo es la inversa multiplicativa de la distancia del camino más corto entre los nodos. La eficiencia global media de un grafo es la eficiencia media de todos los pares de nodos [24]. Medir la eficiencia de cada red que se va creando tras la sucesión de ataques es interesante para conocer precisamente la progresión de la eficiencia de la red. A lo largo del documento se denota como $E(k)$ la eficiencia de la red tras la eliminación de k nodos. De nuevo se utiliza el valor relativo:

$$\mathcal{E}(k) = \frac{E(k)}{E(0)} \tag{3.3}$$

En este punto, cabe mencionar que se valoró utilizar la conectividad algebraica como métrica local, y se hicieron varias pruebas obteniendo distintos resultados. La conectividad algebraica se define como el segundo valor propio más pequeño $a(G)$ de la matriz de adyacencia binaria $A(G)$, que es la matriz cuadrada formada a partir del grafo G [25]. Sin embargo, al igual que los autores *S. Xiaoqian et al.* en Ref. [12], se ha detectado que el gráfico resultante del estudio de la conectividad algebraica no era monótonamente decreciente. Es por ello que, se considera que la conectividad algebraica no aporta resultados relevantes

para este estudio. Es una medida que se podría considerar relevante en ataques a aristas o links, cuando no varía el número de nodos.

3.3.2. Métricas globales de robustez y eficiencia

Para estudiar cómo afecta un ataque dirigido cuando se eliminan todos los nodos de la red, se hace necesaria la utilización de un coeficiente para así poder comparar cómo afectan los distintos ataques a la vulnerabilidad de la red, cuál es más y menos dañino. Estos coeficientes surgen a partir de las métricas locales descritas con anterioridad, por tanto:

- **Coeficiente de robustez R:** dado un orden de ataque a una red con n nodos, el valor de robustez de la red se define como:

$$R = \frac{1}{n}\sum_{k=1}^{n} R(k) \tag{3.4}$$

  El cálculo de R depende del orden de ataque. Un valor bajo de R se corresponde con una red poco robusta o un ataque muy dañino, mientras que un valor alto de R se corresponde con una red robusta o un ataque que provoca poco daño en la conectividad. El valor de R en una estrella S_n es $1/n$ y en la red completa K_n tiende a $1/2$ cuando $n \to \infty$. En cualquier otra red $0 < R < 1/2$.

- **Coeficiente de robustez R^*:** similar al caso anterior, calcularemos el valor de la robustez de la red como:

$$R^* = \frac{1}{n-1}\sum_{k=1}^{n-1} R^*(k) \tag{3.5}$$

  La interpretación de este coeficiente es análoga a la anterior, cuantificando el daño sobre el tráfico. El valor en una estrella S_n es 0 y en la red completa K_n tiende a $1/3$ cuando $n \to \infty$. En cualquier otra red $0 < R^* < 1/3$. Se divide entre $n-1$ debido a que en el último paso de la sucesión de ataques cuando sólo quede un nodo, no quedará ningún link. Por tanto, $SL(n) = 0$.

- **Coeficiente de eficiencia $R_{\mathcal{E}}$:** de manera análoga a los casos anteriores, se calcula el

coeficiente de eficiencia de la red como:

$$R_{\mathcal{E}} = \frac{1}{n-2} \sum_{k=1}^{n-2} \mathcal{E}(k) \tag{3.6}$$

Se interpreta este coeficiente sabiendo que el valor del mismo en la red completa K_n es 1 debido a que es la red más eficiente, con $\mathcal{E}(k) = 1$ para todo k entre 1 y $n-2$. En cualquier otra red $0 < R_{\mathcal{E}} < 1$. Se divide entre $n-2$ ya que el coeficiente no tiene sentido cuando no queda ningún nodo ni cuando hay únicamente un nodo.

Aunque en este trabajo se utilizan los coeficientes R y R^* tanto para ataques sobre la red no pesada como sobre la red con pesos de similitud, ha de tenerse en cuenta que no recogen la información de los pesos de cada nodo, es decir, no tienen en cuenta las ponderaciones para evaluar la robustez. Por ello, en estos coeficientes no se refleja el peso de cada nodo.

Por otro lado, $R_{\mathcal{E}}$ recoge la información de las disimilitudes para redes pesadas, es decir, recoge la información de los pesos cuando estos son de coste, como distancias geográficas u ortodrómicas. Debido a que en este estudio se utiliza un peso de ganancia o similitud, este coeficiente de eficiencia no tendrá sentido para la red ponderada considerada en este trabajo debido a su naturaleza.

Capítulo 4

Análisis y simulación (EATN)

4.1. Propiedades topológicas de la red

En este apartado se presentan los resultados de las propiedades estructurales de la red no pesada.

Tabla 4.1: Métricas globales de la red sin pesos y notaciones (Fuente: Elaboración propia a partir de datos de *Eurostat* [4] procesados con *Python* [5])

Símbolo	Métrica	Ecuación	Valor
n	Número de nodos	-	357
m	Número de links	$\sum_{i,j} \frac{a_{ij}}{2}$	4347
d	Densidad	$\frac{2m}{n(n-1)}$	0,0684
$\langle k \rangle$	Grado medio	$\sum_i \frac{k_i}{n}$	12,1765
D	Diámetro	$max\, d_{ij}$	5
L	Longitud promedio	$\sum_{i \neq j} \frac{d_{ij}}{n(n-1)}$	2,3670
C	Coeficiente de Clustering	$\sum \frac{C_i}{n}$	0,4528
E	Eficiencia	$\sum_{i \neq j} \frac{1/d_{ij}}{n(n-1)}$	0,4643

4.2. Simulación de ataques dirigidos de nodo

Se presentan los resultados de los coeficientes globales de robustez R y R^* y de eficiencia $R_{\mathcal{E}}$, con los que podremos determinar el daño producido por los diferentes ataques, sobre la red pesada y no pesada. No se realizan ataques B ni C en la red pesada, dado que ambos utilizan, para la ordenación de los nodos en los ataques, información sobre distancias, siendo los pesos utilizados de similitud o ganancia en vez de disimilitud o costes. Tampoco se obtiene el coeficiente $R_{\mathcal{E}}$ sobre la red pesada por el mismo motivo, la eficiencia en una red pesada involucra a información pesada con pesos de disimilitud, no tiene sentido su uso con pesos de similitud.

Tabla 4.2: Coeficientes globales R, R^* y $R_{\mathcal{E}}$ para las redes sin y con pesos (Fuente: Elaboración propia a partir de simulaciones realizadas con *Python* [5])

Ataque	Red sin pesos			Red con pesos	
	R	R^*	$R_{\mathcal{E}}$	R	R^*
DN	0.2458	0.0972	0.1798	0.2513	0.1031
DI	0.2147	**0.0911**	0.1585	**0.2181**	**0.0957**
BN	0.2265	0.1005	0.1625	-	-
BI	**0.1892**	0.0997	**0.1381**	-	-
CN	0.2715	0.1064	0.2083	-	-
CI	0.2304	0.0967	0.1780	-	-
EN	0.2784	0.1063	0.2140	0.2794	0.1107
EI	0.2242	0.0928	0.1711	0.2309	0.0980

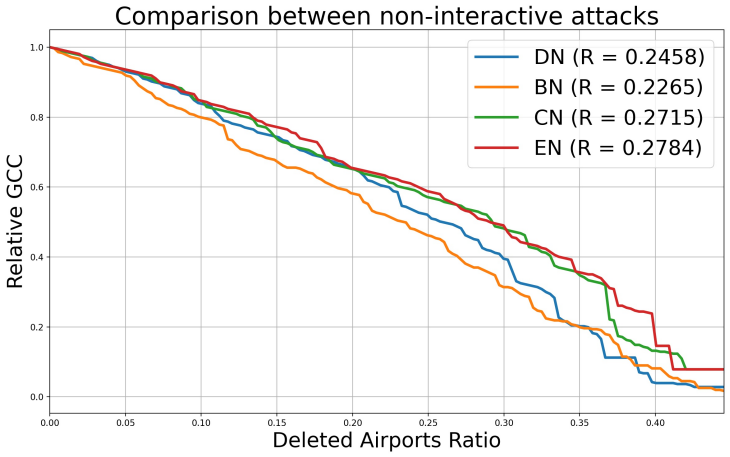
A continuación, en la tabla 4.3 se muestran los 10 primeros aeropuertos eliminados de la red no ponderada para cada ataque.

Tabla 4.3: Primeros 10 aeropuertos eliminados en cada ataque; red no ponderada (Fuente: Elaboración propia a partir de simulaciones realizadas con *Python* [5])

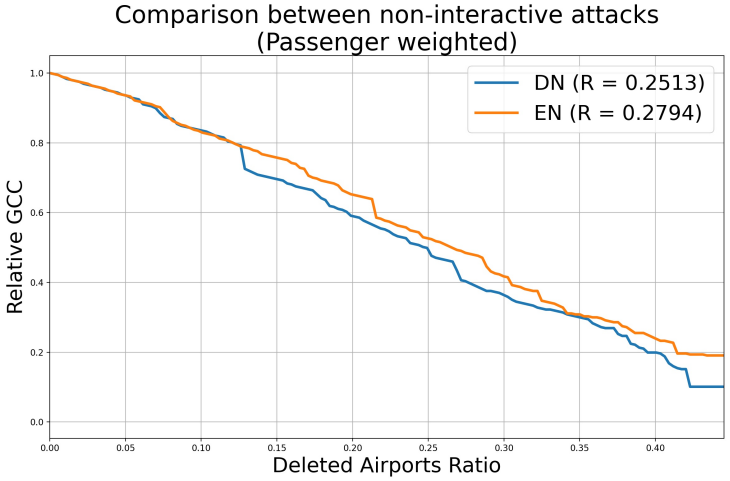
	DN	DI	BN	BI
1	Amsterdam Airport Schiphol	Amsterdam Airport Schiphol	Istanbul Airport	Istanbul Airport
2	Frankfurt am Main Airport	Frankfurt am Main Airport	Stockholm-Arlanda Airport	Sabiha Gökçen International
3	London Stansted Airport	London Stansted Airport	Amsterdam Airport Schiphol	Ataturk International Airport
4	London Gatwick Airport	London Gatwick Airport	Oslo Gardermoen Airport	Esenboğa International Airport
5	Munich Airport	Munich Airport	London Gatwick Airport	Amsterdam Airport Schiphol
6	Manchester Airport	Manchester Airport	London Stansted Airport	Stockholm-Arlanda Airport
7	Dublin Airport	Dublin Airport	Sabiha Gökçen International	Oslo Gardermoen Airport
8	Barcelona International Airport	Barcelona International Airport	Ataturk International Airport	London Gatwick Airport
9	Charles de Gaulle International	Charles de Gaulle International	Charles de Gaulle International	London Stansted Airport
10	Palma De Mallorca Airport	Palma De Mallorca Airport	Frankfurt am Main Airport	Frankfurt am Main Airport
	CN	**CI**	**EN**	**EI**
1	Amsterdam Airport Schiphol	Amsterdam Airport Schiphol	Amsterdam Airport Schiphol	Amsterdam Airport Schiphol
2	Frankfurt am Main Airport	Frankfurt am Main Airport	Frankfurt am Main Airport	Frankfurt am Main Airport
3	London Gatwick Airport	London Gatwick Airport	Munich Airport	Munich Airport
4	Munich Airport	Munich Airport	Dublin Airport	Dublin Airport
5	Dublin Airport	Dublin Airport	Barcelona International Airport	Barcelona International Airport
6	Manchester Airport	Manchester Airport	Manchester Airport	Manchester Airport
7	Barcelona International Airport	Barcelona International Airport	Charles de Gaulle International	Charles de Gaulle International
8	London Stansted Airport	London Stansted Airport	Adolfo Suárez Madrid Barajas	Adolfo Suárez Madrid Barajas
9	Charles de Gaulle International	Charles de Gaulle International	London Gatwick Airport	London Gatwick Airport
10	Adolfo Suárez Madrid Barajas	Adolfo Suárez Madrid Barajas	Copenhagen Kastrup Airport	Copenhagen Kastrup Airport

A continuación, se presentan las figuras correspondientes a los valores de las métricas obtenidos tras la simulación de ataques realizados sobre las redes sin y con pesos.

En las figuras 4.1a y 4.1b se presentan dos gráficos en los que se muestran los ataques no-interactivos para las redes sin y con pesos respectivamente, utilizando la métrica relativa del tamaño de la componente gigante. En la figura 4.1b, correspondiente a la red con pesos, únicamente se muestran los ataques que tienen sentido en redes ponderadas.



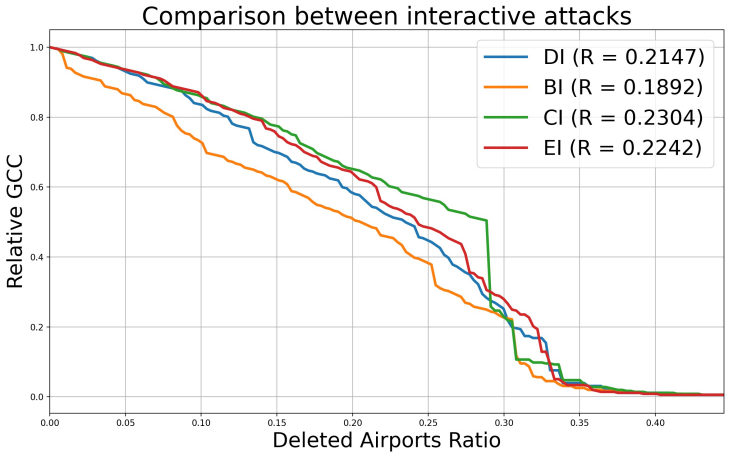
(a) Red sin pesos



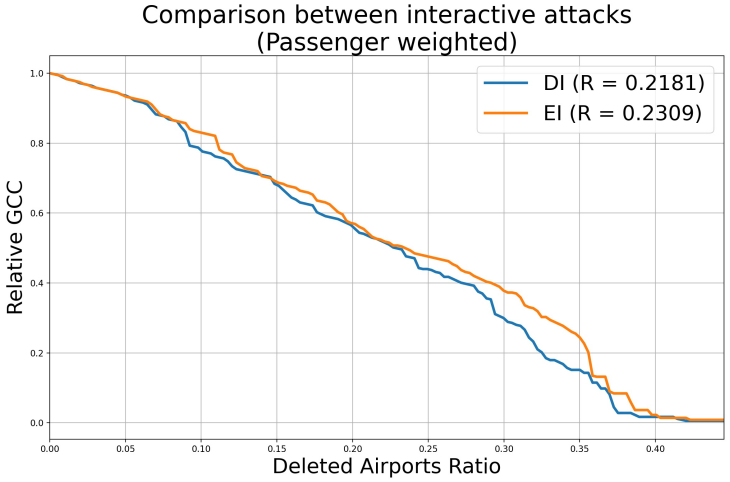
(b) Red con pesos

Figura 4.1: Comparativa entre los ataques no-interactivos; métrica $R(k)$ (Fuente: Elaboración propia)

A continuación, se pasan a presentar las comparativas de ataques interactivos. En las figuras 4.2a y 4.2b se presentan los ataques interactivos para las redes sin y con pesos respectivamente, utilizando la métrica relativa del tamaño de la componente gigante.



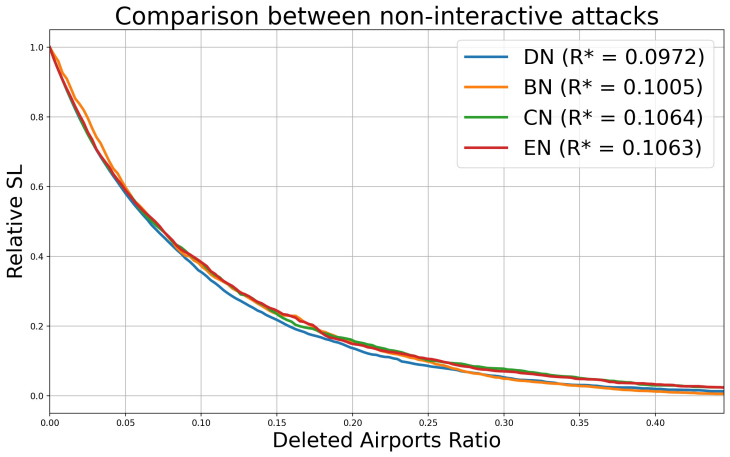
(a) Red sin pesos



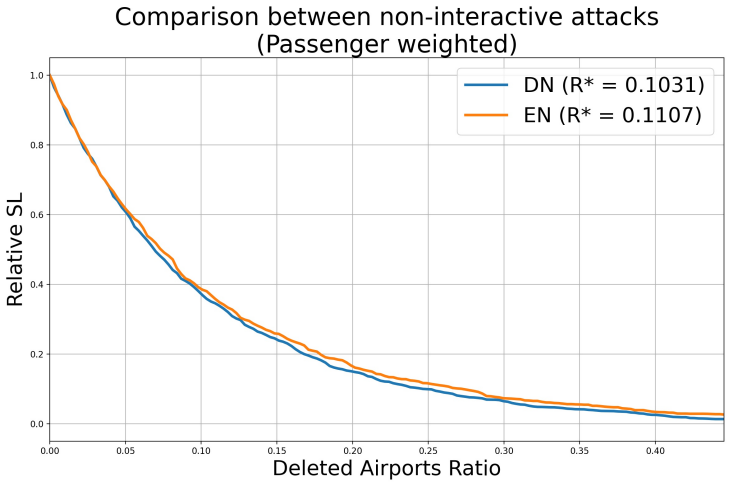
(b) Red con pesos

Figura 4.2: Comparativa entre los ataques interactivos; métrica $R(k)$ (Fuente: Elaboración propia)

En las figuras 4.3a y 4.3b, se muestra una comparativa de los ataques no-interactivos utilizando la métrica relativa de los links que sobreviven.



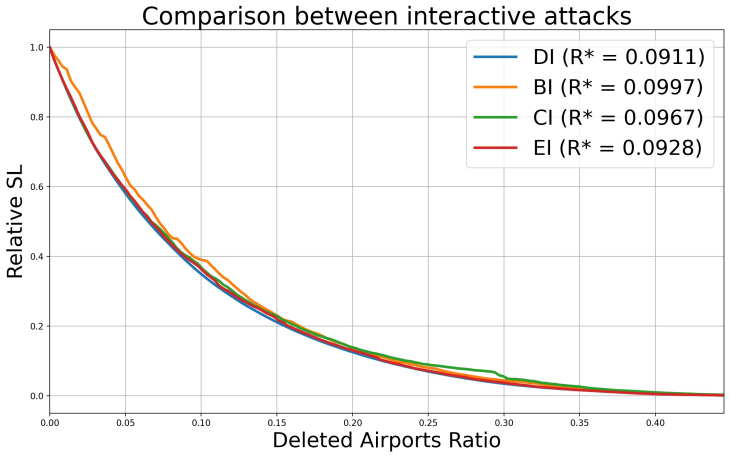
(a) Red sin pesos



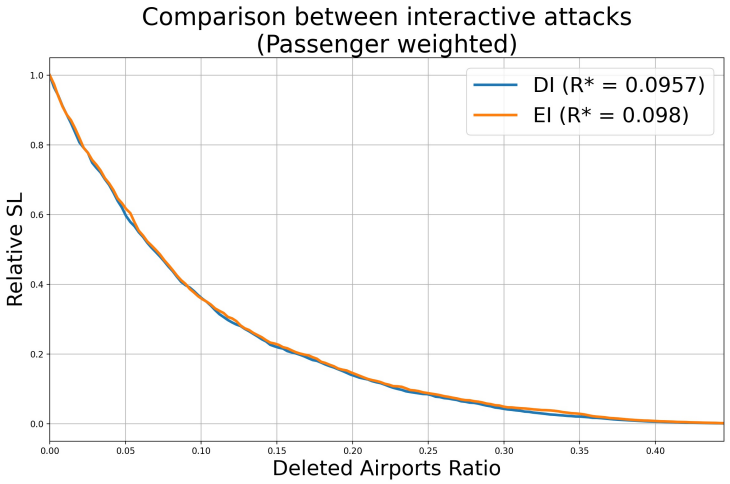
(b) Red con pesos

Figura 4.3: Comparativa entre los ataques no-interactivos; métrica $R^*(k)$ (Fuente: Elaboración propia)

De manera similar, en las figuras 4.4a y 4.4b, se muestra una comparativa de los ataques interactivos utilizando la métrica relativa de los links que sobreviven.



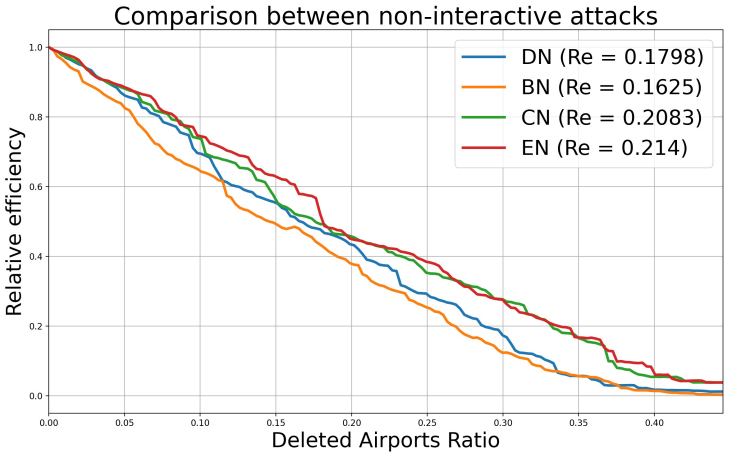
(a) Red sin pesos



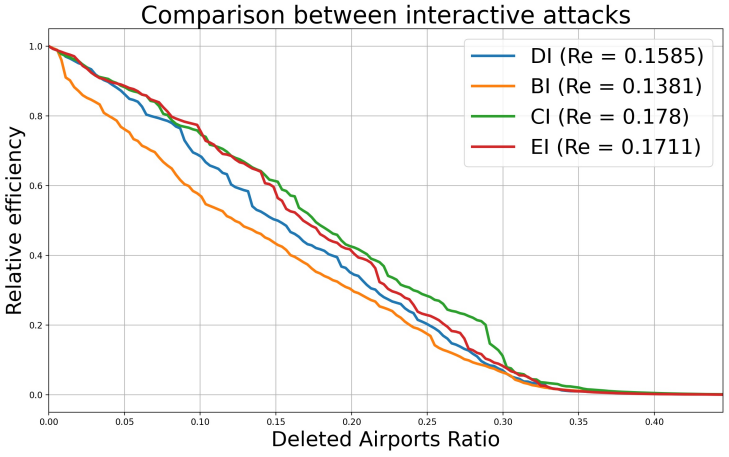
(b) Red con pesos

Figura 4.4: Comparativa entre los ataques interactivos; métrica $R^*(k)$ (Fuente: Elaboración propia)

En la figuras 4.5a y 4.5b, se muestra una comparativa de la eficiencia relativa entre ataques no-interactivos e interactivos, respectivamente, para la red no ponderada.



(a) Red sin pesos



(b) Red sin pesos

Figura 4.5: Comparativa entre los ataques no-interactivos e interactivos; métrica $\mathcal{E}(k)$ (Fuente: Elaboración propia)

Toda vez presentadas los gráficos comparativos entre los ataques no-interactivos y los interactivos, se proponen a continuación, matrices de cuatro gráficos en los que se compara para cada ataque y métrica, la red sin pesos o con pesos y la red completa. Cuando se compara con la red completa, los resultados de realizar sucesivos ataques sobre la misma son

independientes del tipo de ataque realizado y de si la red es o no pesada. Esto se debe a que cada vez que se elimina un nodo, la red resultante es otra red completa con un nodo menos y todos los links posibles. Es por ello que en la red completa pueden incorporarse o no pesos, pero el valor de los pesos es irrelevante, los valores de de $R(k)$ y $R^*(k)$ son los mismos que en la red no pesada.

En las figuras 4.6 y 4.7 se observa la comparativa con la red completa por ataques, incluyendo los no-interactivos y los interactivos de la red sin pesos y con pesos respectivamente. La métrica relativa utilizada es el tamaño de la componente gigante $R(k)$.

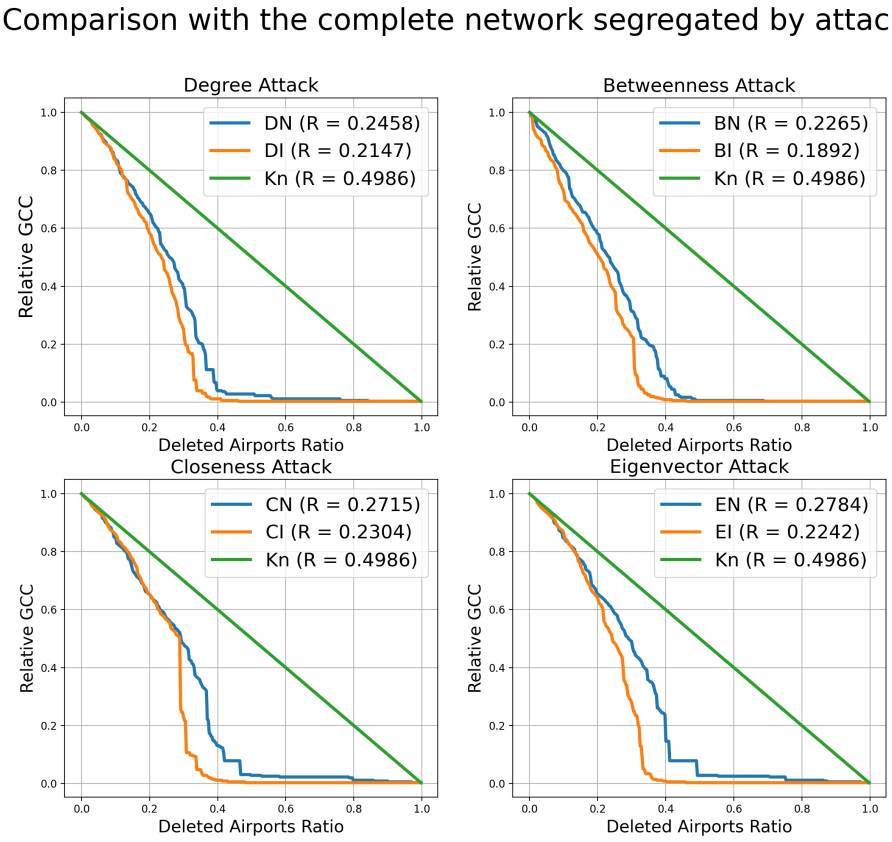


Figura 4.6: Comparativa por ataque de las redes sin pesos y completa; métrica $R(k)$ (Fuente: Elaboración propia)

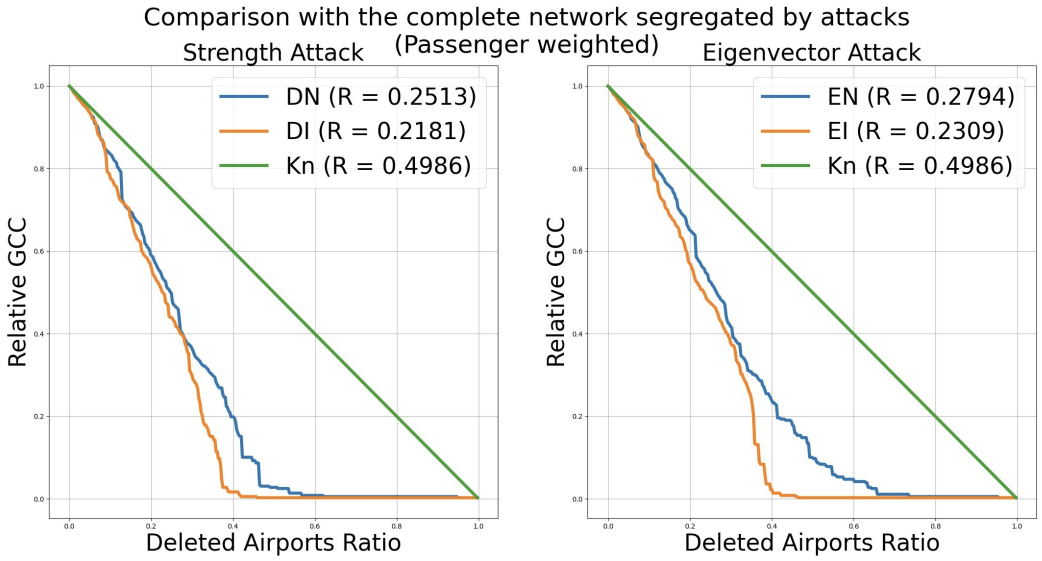


Figura 4.7: Comparativa por ataque de las redes con pesos y completa; métrica $R(k)$ (Fuente: Elaboración propia)

En las figuras 4.8 y 4.9 se observa la comparativa con la red completa utilizando los links que sobreviven $R^*(k)$ como métrica.

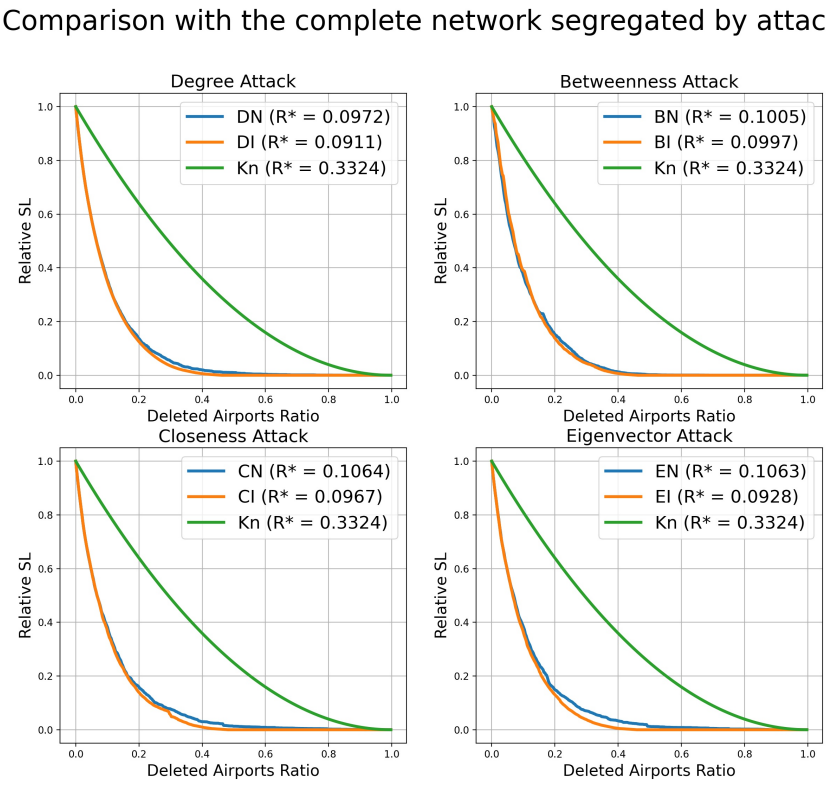


Figura 4.8: Comparativa por ataque de las redes sin pesos y completa; métrica $R^*(k)$ (Fuente: Elaboración propia)

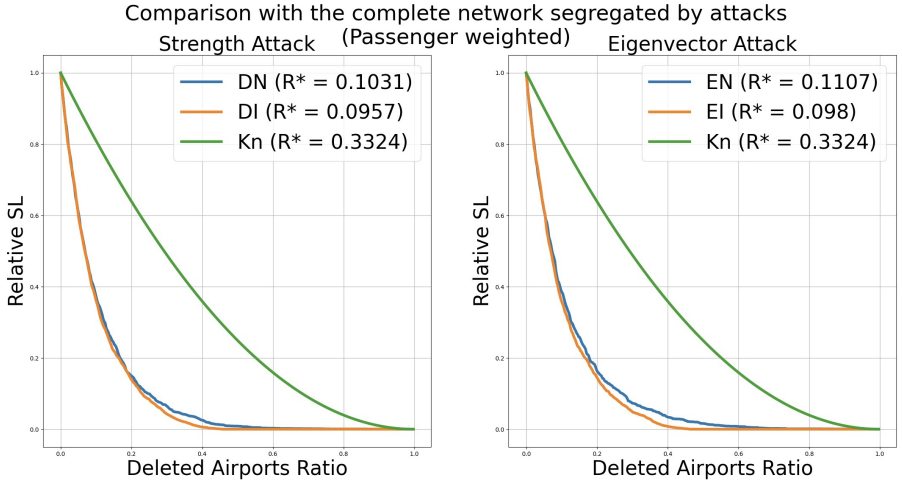


Figura 4.9: Comparativa por ataque de las redes con pesos y completa; métrica $R^*(k)$ (Fuente: Elaboración propia)

Por último, en la figura 4.10 se muestra la comparativa de la red no ponderada con la red completa segregada por ataques utilizando la métrica $\mathcal{E}(k)$.

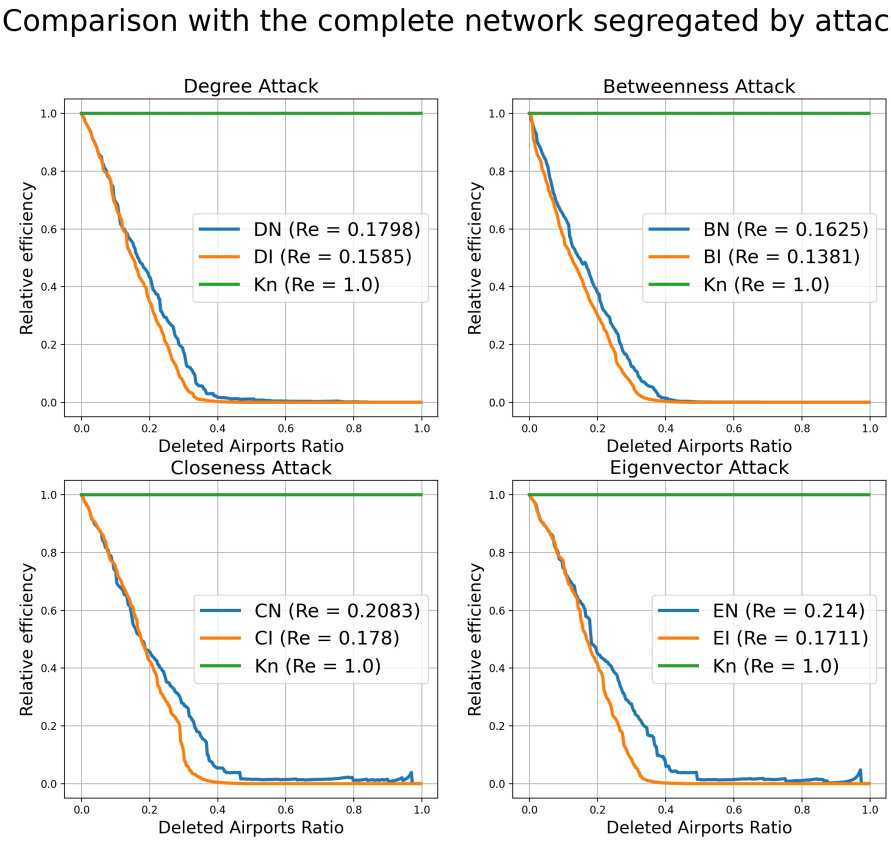


Figura 4.10: Comparativa con K_n por ataque; métrica $\mathcal{E}(k)$ (Fuente: Elaboración propia)

4.3. Discusión

En esta sección se discutirán los resultados presentados en las secciones 4.1 y 4.2.

Inicialmente, observando la tabla 4.1, podemos tener una visión global de la estructura de la red. La densidad de la red europea tiene un valor bajo ($d_{eu} = 0{,}0684$) comparado con otras redes de transporte aéreo como la española ($d_{es} = 0{,}303$) [9], la italiana ($d_{it} = 0{,}302$) [26] o la estadounidense ($d_{us} = 0{,}178$) [10] que tiene un valor ligeramente inferior a las dos últimas, sin embargo, su valor es cercano a redes de aeropuertos como la china ($d_{ch} = 0{,}099$) [11] y la australiana ($d_{au} = 0{,}069$) [27]. En cuanto al grado medio, la red estudiada tiene un grado medio ($\langle k \rangle_{eu} = 12{,}18$) similar al de otras redes de aeropuertos de la literatura como la red española ($\langle k \rangle_{es} = 11{,}8$) [9], la italiana ($\langle k \rangle_{it} = 12{,}4$) [26], la china ($\langle k \rangle_{ch} = 14{,}14$) [11] o la india ($\langle k \rangle_{in} = 11{,}52$) [28]. Esto quiere decir, por una parte, que la red podría tener significativamente más rutas de las que tiene actualmente, y por otra que el número medio de conexiones directas por aeropuerto es similar al de otras redes.

Los aeropuertos con mayor grado, es decir, los que más conexiones directas tienen con otros aeropuertos europeos, son los de Ámsterdam, Frankfurt, Londres Stansted, Londres Gatwick, Munich, Mánchester, Dublín y Barcelona. Si nos fijamos en los aeropuertos con mayor fuerza, teniendo en cuenta que el peso es el número de pasajeros, éstos son los de Ámsterdam, Madrid Barajas, Barcelona, Frankfurt, París Charles de Gaulle, Londres Heathrow, Munich y Londres Gatwick. Todos los aeropuertos mecionados corresponden a las capitales europeas con mayor tráfico aéreo. Se observa que no tienen por qué coincidir los aeropuertos con mayor grado y con mayor fuerza.

El diámetro de la red es 5 y la longitud promedio es 2,37, lo que quiere decir que no todos los nodos están conectados directamente y que es necesario de media realizar al menos una escala. El máximo número de escalas a realizar para llegar a cualquier nodo de la red es 4. El coeficiente de clustering es 0,45, aunque no es un valor alto, comparado por ejemplo con el de la red española $C_{es} = 0{,}73$ [9], indica cierta tendencia de agrupación de los aeropuertos europeos, que se debe fundamentalmente a las conexiones en cada subred nacional. El valor del coeficiente de clustering parece indicar que la EATN tiene propiedades de red de *mundo*

pequeño.

En cuanto a la eficiencia de EATN ($E = 0{,}46$), es casi un 30 % mayor que la eficiencia de la red mundial $E_{wo} = 0{,}36$ [12] pero menor que la eficiencia de la red española $E_{es} = 0{,}64$ [9].

En la tabla 4.3 donde se muestran los primeros 10 aeropuertos eliminados de la red no ponderada para cada ataque, se puede extraer que BI, el ataque más dañino según R y $R_{\mathcal{E}}$, elimina inicialmente aeropuertos turcos que fueron los que mayor centralidad de intermediación tuvieron en cada paso de la sucesión. Observamos que en el resto de métricas, estos aeropuertos no tienen tanta relevancia como para estar entre los primeros 10 eliminados de la red. Esto indica que los aeropuertos turcos, aunque no son los aeropuertos más relevantes en cuanto a su capacidad de tráfico, sí lo son como nodos de comunicación, siendo importantes como aeropuertos de escala para viajar a otros destinos europeos.

A continuación, se pasa a comentar los resultados correspondientes a los coeficientes globales de robustez y eficiencia, donde se puede determinar qué ataques son más dañinos y sobre qué red.

Tal y como se expresó en la subsección 3.3.2, el peor ataque viene representado por el menor valor del coeficiente global R. Como se puede observar en la parte relativa a la red sin pesos de la tabla 4.2, los ataques interactivos tienen en todos los casos menor valor de R que sus homólogos no interactivos. Esto tiene sentido, ya que en los ataques dirigidos se calcula en cada paso de la sucesión de ataques el aeropuerto o nodo más importante en función del ataque, mientras que en los ataques no-interactivos se realiza un listado de nodos por importancia inicial y se eliminan sucesivamente. Esto puede dar lugar a que, sobre todo en las eliminaciones finales, se eliminen nodos que no son los más importantes en cada caso, porque la red ha cambiado significativamente respecto de la inicial.

Otro aspecto que se puede extraer de la red sin pesos en la tabla 4.2, es que el ataque más dañino ha sido el BI, centralidad de intermediación interactivo. Esta es la misma conclusión a la que llegaron *S. Wandelt et al.* en Ref. [20], donde BI fue el ataque más efectivo en 6 de las 8 redes utilizadas en el estudio. Entre los ataques no-interactivos o estáticos, BN fue también el más dañino sobre la red sin pesos. Esto destaca la importancia de la alta centralidad de intermediación en la conectividad de redes. Se observa que, para la red sin pesos, los ataques

menos efectivos en función del coeficiente han sido CN y EN.

En cuanto al coeficiente R^* para la red sin pesos, relativo a los links o aristas que sobreviven, el ataque más efectivo ha sido el DI, y el menos efectivo el CN junto con el EN, al igual que para el coeficiente R. Se observa también, que los ataques interactivos son más efectivos o dañinos para la red que los ataques no-interactivos.

El coeficiente de eficiencia $R_{\mathcal{E}}$ para la red no ponderada tiene un comportamiento similar al coeficiente R, en el que BI se presenta como el ataque más dañino dado que resulta en un valor para el coeficiente inferior al resto. Los ataques CN y EN son de nuevo los menos dañinos sobre la red.

De nuevo, en la tabla 4.2, en la parte que hace referencia a la red con pesos, podemos observar que, estudiando los valores de R, los ataques interactivos son más efectivos que los ataques no-interactivos. También, se extrae que el ataque más efectivo es el DI. El ataque menos efectivo ha sido EN.

De forma similar para el coeficiente R^* en la red con pesos, los ataques interactivos son los más efectivos, el ataque más efectivo ha sido DI al igual que para la red sin pesos, y el ataque menos efectivo ha sido EN.

Una vez analizados los valores globales de las métricas de robustez y eficiencia, se discuten a continuación los resultados obtenidos con evaluación de las métricas locales $R(k)$, $R^*(k)$ y $R_{\mathcal{E}}(k)$ en cada paso k de los ataques. La evolución de estas métricas se visualiza mediante gráficas comparativas de los diferentes ataques, así como de los valores sobre la red más robusta, es decir, la red completa.

Antes de profundizar en los detalles de cada figura presentada en la sección 4.2, es necesario realizar un análisis global de cómo responde la red EATN ante los ataques perpetrados sobre la misma. Las medidas propuestas, GCC y SL, son muy útiles para observar el estado de la red a medida que se avanza en la sucesión de ataques, ya que GCC nos muestra el número de nodos del mayor subgrafo conectado y SL los links que sobreviven. Si revisamos por ejemplo las figuras 4.1 y 4.2, se observa que $R(k)$ va disminuyendo de tal manera que, cuando se han eliminado el $\approx 30\,\%$ de los nodos, la red colapsa, debido a que $R(k) < 0{,}5$. Lo que contrasta con $R^*(k)$, ya que observando las figuras 4.3 y 4.4, se detecta que la velocidad

de eliminación del número de links es mucho mayor, de tal manera que $R^*(k)$ es menor que 0,5 cuando se eliminan únicamente el $\approx 7\,\%$ de los nodos. La lectura de esto es interesante, ya que cuando se eliminan el $\approx 7\,\%$ de los nodos, el subgrafo conectado no ha disminuido significativamente respecto del inicial, pero la red ha perdido una cantidad significativa de links. Cuando se eliminan el $\approx 30\,\%$ de los nodos de la red, la mayor componente conexa tiene aproximadamente la mitad de los nodos iniciales pero una densidad mucho menor, ya que el número de links es aproximadamente el $10\,\%$ del inicial. A continuación, pasaremos a analizar las figuras presentadas en la sección 4.2.

En la figura 4.1 se observa que $R(k)$ desciende casi linealmente hasta eliminar el $40\,\%$ de los aeropuertos, donde se estabiliza dado que, a partir de este punto, la componente gigante no varía significativamente, pero siendo muy pequeña respecto de la inicial, perdiendo completamente su funcionalidad. En la figura 4.1a correspondiente a la red sin pesos, se observa que CN y EN quedan por encima indicando que son los menos dañinos, y DN y BN se quedan por debajo. En la figura 4.1b se observan los ataques DN y EN, con DN por debajo indicando que es más dañino. Si se comparan ambas figuras, se observa que los ataques son más dañinos sobre la red sin pesos, ya que consiguen disminuir el tamaño de la componente gigante más rápido.

Si pasamos a comparar los ataques interactivos en la figura 4.2, lo primero que se percibe es que consiguen disminuir el tamaño de la componente gigante con mayor rapidez que los anteriores. Esta es una de las conclusiones a las que se llegó cuando se discutieron los resultados de R. Además, se vuelve a observar que los ataques son más efectivos en la red sin pesos que en la red ponderada. En la figura 4.2a, se observa que la diferencia entre BI y el resto viene desde el inicio, es decir, cuando están todos los nodos en la red, BI escoge inicialmente los nodos clave para reducir GCC ligeramente respecto al resto de ataques. Pese a que cuando se eliminen aproximadamente el $40\,\%$ de los nodos el resultado sea parecido, esto es lo que hace que BI tenga un coeficiente R menor que el resto y sea por tanto el más dañino.

En las figuras 4.3 y 4.4, correspondientes al estudio de $R^*(k)$, se saca como conclusión que, en la red sin pesos (figuras 4.3a y 4.4a), todos los ataques eliminan links a un ritmo muy

similar, tal y como demuestra que los coeficientes R^* sean muy parecidos. El estudio de la red con pesos se puede observar en las figuras 4.3b y 4.4b. Se observa también que, cuando se eliminan aproximadamente el 40 % de los aeropuertos, el número de rutas de la red no varía significativamente. Si comparamos estos gráficos con los anteriores, se observa que es mucho más rápido el ritmo de eliminación de rutas que el de disminución de la componente gigante, pues mientras en GCC se tiene una tendencia decreciente lineal, en SL se muestra un decrecimiento mayor.

En cuanto a la eficiencia relativa, mostrada en la figura 4.5, para la red sin pesos (figuras 4.5a y 4.5b) se vuelve a observar que BI marca una diferencia desde el inicio de la sucesión de ataques. De nuevo, los ataques interactivos son más eficientes que los no interactivos, y la eficiencia no tiene variaciones significativas a partir de la eliminación de aproximadamente el 40 % de los nodos.

Por último, se pasa a analizar los gráficos en los que se comparan, para los distintos ataques, la opción interactiva, la no-interactiva y la red completa.

En las figuras 4.6 y 4.7 observamos que el ataque producido sobre la red completa es un decrecimiento lineal de $(0, 1)$ a $(1, 0)$ con un valor de R que tiende a $1/2$ cuando $n \to \infty$, como ya se comentó en la subsección 3.3.2. Los ataques no-interactivos, tienen una curva que se acerca más a la de la red completa ya que son menos dañinos que los interactivos.

Observando las figuras 4.8 y 4.9, se tiene que no hay tanta diferencia como anteriormente entre los ataques estáticos e interactivos, lo que reafirma la conclusión a la que se llega anteriormente, de que todos los ataques, independientemente del tipo o de si son estáticos o interactivos, eliminan los links de la red a un ritmo muy similar. Esto es diferente para $R(k)$ y para $\mathcal{E}(k)$, donde existen mayores diferencias entre los ataques.

Por último, si nos fijamos en la figura 4.10, la eficiencia relativa de la red completa es constante y máxima, es decir, 1. En dicha figura se pueden observar fenómenos, ligeramente más marcados en CN y EN, en los que, casi al final de la sucesión de ataques, cuando se eliminan aproximadamente el 90 % de los aeropuertos, se tienen ligeros repuntes de la eficiencia relativa. Esto quiere decir que la red que queda, con un nodo menos, es más efectiva que la anterior. Esto se produce porque el ataque no ha eliminado el nodo más importante y

por tanto no ha conseguido disminuir la eficiencia de la red, sino que ha aumentado la misma. Se produce sobre todo en CN y EN, que son ataques no interactivos, y tienen problemas para ordenar los últimos nodos de la red cuando se tienen todos inicialmente. Los ataques interactivos no presentan este tipo de problema porque reordenan los nodos en cada paso de la sucesión, entonces es más fácil en este tipo de ataques dar con los nodos clave para disminuir la eficiencia de la red.

4.4. Conclusiones

En esta sección se comentan las conclusiones del presente documento.

La red EATN responde a los ataques de tal manera que, a nivel de conectividad, colapsa cuando se eliminan un $\approx 30\,\%$ los aeropuertos, y a nivel de capacidad lo hace cuando se eliminan un $\approx 7\,\%$. Esto lleva a la conclusión de que la red pierde densidad rápidamente ya que la eliminación de links es mucho más rápida que la de nodos.

Las métricas utilizadas están relacionadas entre sí y evalúan de manera precisa la conectividad, capacidad y eficiencia de la red, con unas tendencias similares en todas las métricas estudiadas.

El ataque más dañino sobre la red es BI, aunque los links sufren un daño mayor bajo DI. Ambos ataques son los que causan más daño sobre la red, y consiguen disminuir más rápido las métricas de robustez y eficiencia estudiadas. Esto concuerda con la literatura. Además, los ataques interactivos, debido al hecho de que se recalculan en cada paso de la sucesión de ataques, son mucho más dañinos.

El ataque de grado tiene gran probabilidad de sufrir empates, que se resuelven aleatoriamente o, en nuestro caso, alfabéticamente, perdiendo capacidad para hacer daño. En los otros ataques, esto sucede cuando las medidas de centralidad son 0, momento en el que se tienen varios nodos empatados. Se observa este fenómeno en B, C y E, no siendo capaces de disminuir métricas como la eficiencia sino que la aumentan.

Las medidas de robustez propuestas no utilizan información sobre los pesos, por tanto, no recogen la diferencia entre los ataques a la red ponderada respecto de la no ponderada,

es por ello que se realiza la revisión de las métricas de la red pesada, pero no es posible su comparación con los obtenidos para la red sin pesos.

4.5. Futuro trabajo

En esta sección, se detallan las posibilidades de mejora que presenta el trabajo realizado en este documento.

En primer lugar, y desde el punto de vista del atacante, durante la realización de las simulaciones, la resolución de los empates en los ataques se realiza de forma aleatoria. La implementación de *ataques mixtos* que utilicen más información para deshacer empates podría generar más daño sobre las redes. Se propone combinar ataques D y B y simular estos ataques sobre la red EATN y otras redes de transporte o redes simuladas para cuantificar el daño producido y compararlo con los ataques no mixtos.

En segundo lugar, el estudio de robustez y eficiencia podría mejorarse incorporando pesos de disimilitud como pueden ser distancias ortodrómicas o tiempos de vuelo y tiempos de escala para implementar ataques B y C sobre la red pesada y valorar la eficiencia en base a estas disimilitudes. Del mismo modo, se ha comprobado que para pesos de similitud, los ataques D y E utilizan la información de tráfico de los nodos pero los coeficientes R y R^* no son capaces de recoger esta información, por lo que se propone estudiar generalizaciones para estos coeficientes que utilicen los pesos para mejorar la valoración del daño.

También se podrían utilizar otras estrategias de ataque como la ordenación de los nodos usando la centralidad de Bonacich y realizar ataques sobre links críticos en lugar de sobre los nodos. Además, se pueden incluir otras métricas de robustez en las que se tenga en cuenta los pasajeros no afectados por la eliminación de nodos debido a la modificación de su ruta.

Finalmente, desde el punto de vista de la defensa de las infraestructuras críticas, los resultados obtenidos y las diferentes métricas pueden utilizarse para proponer estrategias de mejora sobre la red que la hagan más robusta frente a ataques dirigidos.

Referencias bibliográficas

[1] Wolfram Research, "Wolfram mathworld." https://mathworld.wolfram.com. Acceso: 21/06/2022.

[2] P. Johnson, "Graph theory," *Github Pages*, 2018.

[3] K. Kumar, "What is merge sort algorithm: How does it work, its advantages and disadvantages." https://www.simplilearn.com. Acceso: 27/06/2022.

[4] Comisión Europea, "Eurostat - Oficina Europea de Estadística, oficina estadística de la comisión europea." https://ec.europa.eu/eurostat/. Acceso: 18/06/2022.

[5] G. van Rossum, "Python." https://www.python.org. Acceso: 18/06/2022.

[6] I. Savage, "Comparing the fatality risks in united states transportation across modes and over time," *Research in transportation economics*, vol. 43, no. 1, 2013.

[7] L. E. Rocha, "Dynamics of air transport networks: A review from a complex systems perspective," *Chinese Journal of Aeronautics*, vol. 30, no. 2, 2017.

[8] S. M. Wilkinson, S. Dunn, and S. Ma, "The vulnerability of the european air traffic network to spatial hazards," *Natural hazards*, vol. 60, no. 3, 2012.

[9] M. Trobajo de las Matas and M. Carriegos Vieira, "Spanish Airport Network structure: Topological characterization," *Computational and Mathematical Methods*, 2022.

[10] Z. Xu and R. Harriss, "Exploring the structure of the US intercity passenger air transportation network: A weighted complex network approach," *GeoJournal*, vol. 73, no. 2, 2008.

[11] J. Wang, H. Mo, F. Wang, and F. Jin, "Exploring the network structure and nodal centrality of China's air transport network: A complex network approach," *Journal of Transport Geography*, vol. 19, no. 4, 2011.

[12] X. Sun, V. Gollnick, and S. Wandelt, "Robustness analysis metrics for worldwide airport network: A comprehensive study," *Chinese Journal of Aeronautics*, vol. 30, no. 2, 2017.

[13] H. A. M. Malik, N. Mahmood, M. H. Usman, and F. Abid, "Un-weighted network study of pakistani airports," in *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, IEEE, 2019.

[14] E. P. García del Valle, G. Lagunes García, L. Prieto Santamaría, M. Zanin, E. Menasalvas Ruiz, and A. Rodríguez-González, "Disease networks and their contribution to disease understanding: A review of their evolution, techniques and data sources," *Journal of Biomedical Informatics*, vol. 94, 2019.

[15] M. Trobajo de las Matas, J. Cifuentes Rodríguez, and M. Carriegos Vieira, "On dynamic network security: A random decentering algorithm on graphs," *Open Mathematics*, vol. 16, no. 1, 2018.

[16] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, 1978.

[17] U. Brandes and D. Fleischer, "Centrality measures based on current flow," in *Annual symposium on theoretical aspects of computer science*, Springer, 2005.

[18] S. Wasserman, K. Faust, *et al.*, "Social network analysis: Methods and applications," *Cambridge university press*, 1994.

[19] D. J. Watts and S. H. Strogatz, "Collective dynamics of *small-world* networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[20] S. Wandelt, X. Shi, and X. Sun, "Estimation and improvement of transportation network robustness by exploiting communities," *Reliability Engineering & System Safety*, vol. 206, 2021.

[21] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using Networkx," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), 2008.

[22] S. Segarra and A. Ribeiro, "Stability and continuity of centrality measures in weighted graphs," *IEEE Transactions on Signal Processing*, vol. 64, no. 3, 2015.

[23] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han, "Attack vulnerability of complex networks," *Physical review E*, vol. 65, no. 5, 2002.

[24] V. Latora and M. Marchiori, "Efficient behavior of small-world networks," *Physical review letters*, vol. 87, no. 19, 2001.

[25] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, vol. 23, no. 2, 1973.

[26] M. Guida and F. Maria, "Topology of the Italian airport network: A scale-free small-world network with a fractal structure?," *Chaos, Solitons & Fractals*, vol. 31, no. 3, 2007.

[27] M. M. Hossain and S. Alam, "A complex network approach towards modeling and analysis of the Australian Airport Network," *Journal of Air Transport Management*, vol. 60, 2017.

[28] G. Bagler, "Analysis of the airport network of India as a complex weighted network," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 12, 2008.

Anexo A

Código de *Python*

En este anexo se incluye el código en *Python* correspondiente a la implementación de los ataques realizados sobre la red.

Código A.1: Implementación de los ataques

```
from turtle import color
import numpy as np
import pandas as pd
import networkx as nx
import math
import matplotlib.pyplot as plt

def ordena_degree(df, path, attack):
    '''
    Función para calcular el ataque seleccionado y ordenar de mayor a menor los
    ↪ aeropuertos
    Inputs:
        - df: dataframe con los aeropuertos y las rutas
        - path: nombre del archivo importado para detectar si hay o no pesos
        - attack: tipo de ataque elegido
    Outputs:
        - airports_sorted: listado de aeropuertos ordenados según el ataque seleccionado
    '''
```

```
18
19      # Llamamos a la función creategraph para crear el grafo
20      G = creategraph(df, path)
21      # Extraemos el listado de aeropuertos del dataframe que son los índices del mismo
22      airports = df.index.values.tolist()
23      # Llamamos a la función decidepesos para saber si la matriz es con o sin pesos
24      a = decidepesos(path)
25      # Si a = True la matriz es con pesos
26      if a:
27          w = "weight"
28      else:
29          w = None
30
31      # Si el ataque es "degree" entra en el condicional 1
32      if attack == "degree":
33          # Creamos un vector de ceros para almacenar el "degree" de cada aeropuerto
34          k = np.zeros(len(G))
35          for i in range(len(G)):
36              # Utilizamos la función .degree del paquete networkx para calcular el
                ↪  "degree" y lo almacenamos en k
37              k[i] = G.degree(airports[i], weight=w)
38          # Creamos un dataframe con la columna grado y con los aeropuertos como índices
39          df_k = pd.DataFrame(k, index=airports, columns=["degree"])
40
41      # Si el ataque es "betweenness" entra en el condicional 2
42      elif attack == "betweenness":
43          # Utilizamos la función del paquete networkx para calcular la centralidad
            ↪  "betweenness"
44          b = nx.betweenness_centrality(G, k=None, normalized=True, weight=w,
            ↪  endpoints=True, seed=None)
45          # Creamos un dataframe con los valores resultado y con los aeropuertos como
            ↪  índices
46          df_k = pd.DataFrame.from_dict(b, orient="index", dtype=None, columns=["degree"])
47
48      # Si el ataque es "closeness" entra en el condicional 3
```

```
49      elif attack == "closeness":
50          # Utilizamos la función del paquete networkx para calcular la centralidad
            ↪ "closeness"
51          b = nx.closeness_centrality(G, u=None, distance=None, wf_improved=True)
52          # Creamos un dataframe con los valores resultado y con los aeropuertos como
            ↪ índices
53          df_k = pd.DataFrame.from_dict(b, orient="index", dtype=None, columns=["degree"])
54
55      # Si el ataque es "eigenvector" entra en el condicional 4
56      elif attack == "eigenvector":
57          # Utilizamos la función del paquete networkx para calcular la centralidad
            ↪ "eigenvector"
58          b = nx.eigenvector_centrality(G, max_iter=100000, tol=1e-03, nstart=None,
            ↪ weight=w)
59          # Creamos un dataframe con los valores resultado y con los aeropuertos como
            ↪ índices
60          df_k = pd.DataFrame.from_dict(b, orient="index", dtype=None, columns=["degree"])
61
62      # Si no ha entrado en ninguno de los anteriores mostramos un mensaje de error
63      else:
64          print("Error en la introducción del ataque")
65
66      # Ordenamos el dataframe utilizando los valores resultado y de mayor a menor
67      # quicksort es determinista e inestable
68      # mergesort es determinista y estable
69      maxtomin = df_k.sort_values(by ="degree", ascending=False, kind="mergesort")
70      # Creamos una lista con los aeropuertos ordenados como resultado
71      airports_sorted = maxtomin.index.values.tolist()
72
73      return airports_sorted
74
75  def creategraph(df, path):
76      '''
77      Función para crear un grafo a partir de un dataframe
78      Inputs:
```

```
79          - df: dataframe a utilizar para crear el grafo
80          - path: nombre del archivo importado para detectar si hay o no pesos
81      Outputs:
82          - G: grafo creado
83      '''
84
85      # Extraemos el listado de aeropuertos del dataframe, serán los índices del mismo
86      airports = df.index.values.tolist()
87      # Creamos una lista vacía para almacenar las rutas
88      edges = []
89      # Realizamos un bucle para crear las rutas, que almacenaremos en la lista creada
90      for i in range(len(df)):
91          for j in range(len(df)):
92              if math.isnan(df[airports[i]][airports[j]]) == False:
93                  # Si el valor del dataframe no es un nan, añadimos la ruta a la lista
94                  # El paquete networkx requiere poner las rutas de la forma (origen,
                    ↪  destino)
95                  edges.append((airports[i], airports[j]))
96
97      # Creamos G como un grafo no dirigido
98      G = nx.Graph()
99      # Añadimos los nodos del grafo que serán los aeropuertos
100     G.add_nodes_from(airports)
101     # Añadimos las rutas extraídas del dataframe anteriormente
102     G.add_edges_from(edges)
103
104     # Llamamos a la función que nos dice si hay o no pesos
105     a = decidepesos(path)
106
107     # En caso de que haya pesos creamos un bucle para introducirlos en el grafo
108     if a:
109         for i in range(len(df)):
110             for j in range(len(df)):
111                 if math.isnan(df[airports[i]][airports[j]]) == False:
112                     # En donde haya datos, es decir, no haya nans, extraeremos el peso
```

```
113                     # Para introducirlo en el grafo, indicamos el origen, el destino y el
                        ↪  tipo de dato
114                     G[airports[i]][airports[j]]["weight"] = df[airports[i]][airports[j]]
115
116     return G
117
118 def get_gicomp(df, path):
119     '''
120     Función para obtener la componente gigante de un grafo dado a partir de un dataframe
121     Inputs:
122         - df: dataframe a utilizar para crear el grafo
123         - path: nombre del archivo importado para detectar si hay o no pesos
124     Outputs:
125         - giantC: componente gigante del grafo dado
126     '''
127
128     # Llamamos a la función para crear un grafo
129     G = creategraph(df, path)
130
131     # Utilizamos el comando del paquete networkx para extraer los subgrafos conectados y
        ↪  en concreto el mayor
132     Gcc = max(nx.connected_components(G), key=len)
133     # Con el mayor subgrafo extraído anteriormente creamos un subgrafo
134     gC = G.subgraph(Gcc)
135     # La componente gigante será el número de nodos de este subgrafo
136     giantC = len(gC.nodes)
137
138     return giantC
139
140 def get_eff(df, path):
141     '''
142     Función para obtener la eficiencia de un grafo dado a partir de un dataframe
143     Inputs:
144         - df: dataframe a utilizar para crear el grafo
145         - path: nombre del archivo importado para detectar si hay o no pesos
```

```
146     Outputs:
147         - geff: eficiencia del grafo dado
148     '''
149 
150     # Llamamos a la función para crear un grafo
151     G = creategraph(df, path)
152     # Obtenemos la eficiencia del grafo
153     geff = nx.global_efficiency(G)
154 
155     return geff
156 
157 def get_algebraic_connectivity(df, path):
158     '''
159     Función para obtener la conectividad algebraica de un grafo dado a partir de un
    ↪  dataframe
160     Inputs:
161         - df: dataframe a utilizar para crear el grafo
162         - path: nombre del archivo importado para detectar si hay o no pesos
163     Outputs:
164         - alg: conectividad algebraica del grafo dado
165     '''
166 
167     G = creategraph(df, path)
168     Gcc = max(nx.connected_components(G), key=len)
169     gC = G.subgraph(Gcc)
170 
171     a = decidepesos(path)
172     if a:
173         w = "weight"
174     else:
175         w = None
176 
177     alg = nx.algebraic_connectivity(gC, weight=w, normalized=False, tol=1e-08,
    ↪  method="tracemin_lu", seed=None)
178 
```

```
179     return alg
180 
181 def drawgraph(G):
182     '''
183     Función para dibujar un grafo
184     Inputs:
185         - G: grafo a dibujar
186     '''
187     pos = nx.spring_layout(G)
188     nx.draw_networkx_nodes(G, pos, cmap=plt.get_cmap('jet'), node_size = 500,
        ↪  node_color="deepskyblue")
189     nx.draw_networkx_labels(G, pos)
190     nx.draw_networkx_edges(G, pos, edge_color='r', arrows=False)
191     plt.show()
192 
193     return
194 
195 def get_index(airports, airport):
196     '''
197     Función para obtener el índice en el que se encuentra un aeropuerto en una lista
198     Inputs:
199         - airports: listado de aeropuertos
200         - airport: aeropuerto a buscar
201     Outputs:
202         - flag: índice en el que se encuentra el aeropuerto dentro de la lista
203     '''
204 
205     for i in range(len(airports)):
206         if airports[i] == airport:
207             flag = i
208 
209     return flag
210 
211 def delete_airport(df, first):
212     '''
```

```
213     Función para eliminar un aeropuerto de un dataframe dado
214     Inputs:
215         - df: dataframe a utilizar para crear el grafo
216         - first: aeropuerto a eliminar del dataframe
217     Outputs:
218         - df: dataframe con el aeropuerto eliminado (fila y columna)
219     '''
220
221     # Extraemos el listado de aeropuertos inicial que serán los índices del dataframe
222     airports_ini = df.index.values.tolist()
223
224     # Utilizando la función definida extraemos el índice en el que se encuentra el
        ↪  aeropuerto a eliminar
225     a = get_index(airports_ini, first)
226
227     # Eliminamos la fila y la columna correspondientes al aeropuerto del dataframe
228     df.drop([airports_ini[a]], axis = 0, inplace = True)
229     df.drop([airports_ini[a]], axis = 1, inplace = True)
230
231     return df
232
233 def decidepesos(path):
234     '''
235     Función para detectar si el archivo importado contiene o no los pesos
236     Inputs:
237         - path: nombre del archivo importado para detectar si hay o no pesos
238     Outputs:
239         - a: variable booleana, si True hay pesos y si False no los hay
240     '''
241
242     # Extraemos del nombre del archivo la parte donde se indica si hay o no pesos
243     if path[5:7] == "sp" or path[5:7] == "fu":
244         a = False
245     else:
246         a = True
```

```
247
248     return a
249
250 def get_R_gcc(vector):
251     '''
252     Función para obtener el valor de la robustez de una red bajo un ataque determinado
    ↪  dividiendo entre N
253     Inputs:
254         - vector: vector que contendrá el tamaño de la componente gigante o los links
    ↪  sobrevividos tras los sucesivos ataques
255     Outputs:
256         - R: valor de la robustez de la red
257         - vector_c: vector relativo de las medidas en el vector inicial
258     '''
259
260     vector_c = np.zeros(len(vector))
261
262     for i in range(len(vector)):
263         vector_c[i] = vector[i]/vector[0]
264
265     R = (np.sum(vector_c) - 1) / len(vector)
266
267     return R, vector_c
268
269 def get_R_sls(vector):
270     '''
271     Función para obtener el valor de la robustez de una red bajo un ataque determinado
    ↪  dividiendo entre N-1
272     Inputs:
273         - vector: vector que contendrá el tamaño de la componente gigante o los links
    ↪  sobrevividos tras los sucesivos ataques
274     Outputs:
275         - R: valor de la robustez de la red
276         - vector_c: vector relativo de las medidas en el vector inicial
277     '''
```

```
278
279     vector_c = np.zeros(len(vector))
280
281     for i in range(len(vector)):
282         vector_c[i] = vector[i]/vector[0]
283
284     R = (np.sum(vector_c) - 1) / (len(vector)-1)
285
286     return R, vector_c
287
288 def get_R_eff(vector):
289     '''
290     Función para obtener el valor de la eficiencia de una red bajo un ataque determinado
    ↪  dividiendo entre N-2
291     Inputs:
292         - vector: vector que contendrá la eficiencia tras los sucesivos ataques
293     Outputs:
294         - R: valor de eficiencia de la red
295         - vector_c: vector relativo de las medidas en el vector inicial
296     '''
297
298     vector_c = np.zeros(len(vector))
299
300     for i in range(len(vector)):
301         vector_c[i] = vector[i]/vector[0]
302
303     R = (np.sum(vector_c) - 1) / (len(vector)-2)
304
305     return R, vector_c
306
307 def funcfinal(path, df, attack, iterativo):
308     '''
309     Función "final" que utiliza las funciones anteriores para una vez se determina un
    ↪  ataque ir
```

```
310     eliminando aeropuertos y calcular la componente gigante, los links que sobreviven y
    ↪   la eficiencia
311     de los grafos que van quedando
312     Inputs:
313         - path: nombre del archivo importado para detectar si hay o no pesos
314         - df: dataframe a utilizar para crear el grafo
315         - attack: variable que contiene el nombre del ataque elegido
316         - iterativo: variable que contiene "SI" o "NO" dependiendo si queremos recalcular
    ↪   o no el ataque
317     Outputs:
318         - gcc: vector con las componentes gigantes de los grafos calculados
319         - sls: vector con los links que sobreviven de los grafos calculados
320         - eff: vector con las eficiencias de los grafos calculados
321         - attacked: listado de aeropuertos eliminados ordenados
322     '''
323
324     # Si la opción iterativa está activada, deberemos recalcular el ataque en cada bucle
325     if iterativo == "SI":
326         # Damos un valor inicial a las iteraciones a realizar para crear los vectores
327         iter = int(len(df))
328         # Iniciamos una lista vacía para almacenar las listas de aeropuertos
329         airports = []
330         # Iniciamos una lista vacía para almacenar el aeropuerto atacado en cada bucle
331         attacked = []
332         # Iniciamos un vector de ceros para almacenar las componentes gigantes en cada
            ↪  bucle
333         gcc = np.zeros(iter)
334         # Iniciamos un vector de ceros para almacenar el número de links que sobreviven
            ↪  en cada bucle
335         sls = np.zeros(iter)
336         # Iniciamos un vector de ceros para almacenar la eficiencia en cada bucle
337         eff = np.zeros(iter)
338         for i in range(iter):
339             # Utilizando la función anterior, obtenemos el listado ordenado de
                ↪  aeropuertos ordenados según
```

```
340             # el ataque seleccionado
341             # Esto se realiza dentro del bucle porque estamos en la opción interactiva
342             airports_sorted = ordena_degree(df, path, attack)
343             # Guardamos en una variable el primer aeropuerto (que habrá que eliminar)
344             first = airports_sorted[0]
345             # Obtenemos la componente gigante del grafo
346             gicomp = get_gicomp(df, path)
347             # Calculamos los links que sobreviven
348             sl = np.sum(df.count())
349             # Calculamos la eficiencia del grafo
350             geff = get_eff(df, path)
351             # Creamos un dataframe nuevo en el que no esté el aeropuerto seleccionado
                ↪  anteriormente
352             df_new = delete_airport(df, first)
353             # Nuestro dataframe ahora será sin el aeropuerto
354             df = df_new
355             # Añadimos el listado de aeropuertos ordenado a la lista creada anteriormente
356             airports.append(airports_sorted)
357             # Añadimos el aeropuerto eliminado a la lista creada anteriormente
358             attacked.append(first)
359             # Añadimos la componente gigante al vector de componentes gigantes
360             gcc[i] = gicomp
361             # Añadimos el número de links sobrevividos al vector de survived links
362             sls[i] = sl
363             # Añadimos la eficiencia al vector de eficiencias
364             eff[i] = geff
365         return gcc, sls, eff, attacked
366
367     elif iterativo == "NO":
368         # Damos un valor inicial a las iteraciones a realizar para crear los vectores
369         iter = int(len(df))
370         # Iniciamos un vector de ceros para almacenar las componentes gigantes en cada
            ↪  bucle
371         gcc = np.zeros(iter)
```

```
372         # Iniciamos un vector de ceros para almacenar el número de links que sobreviven
            ↪ en cada bucle
373         sls = np.zeros(iter)
374         # Iniciamos un vector de ceros para almacenar la eficiencia en cada bucle
375         eff = np.zeros(iter)
376         # Iniciamos una lista vacía para almacenar el aeropuerto atacado en cada bucle
377         attacked = []
378         # Utilizando la función anterior, obtenemos el listado ordenado de aeropuertos
            ↪ ordenados según
379         # el ataque seleccionado
380         # Esto se realiza fuera del bucle porque estamos en la opción no interactiva
381         airports_sorted = ordena_degree(df, path, attack)
382         for i in range(iter):
383             G = creategraph(df, path)
384             drawgraph(G)
385             # Guardamos en una variable el primer aeropuerto (que habrá que eliminar)
386             first = airports_sorted[i]
387             # Obtenemos la componente gigante del grafo
388             gicomp = get_gicomp(df, path)
389             # Calculamos los links que sobreviven
390             sl = np.sum(df.count())
391             # Calculamos la eficiencia del grafo
392             geff = get_eff(df, path)
393             # Creamos un dataframe nuevo en el que no esté el aeropuerto seleccionado
                ↪ anteriormente
394             df_new = delete_airport(df, first)
395             # Nuestro dataframe ahora será sin el aeropuerto
396             df = df_new
397             # Añadimos el aeropuerto eliminado a la lista creada anteriormente
398             attacked.append(first)
399             # Añadimos la componente gigante al vector de componentes gigantes
400             gcc[i] = gicomp
401             # Añadimos el número de links sobrevividos al vector de survived links
402             sls[i] = sl
403             # Añadimos la eficiencia al vector de eficiencias
```

```
404             eff[i] = geff
405         return gcc, sls, eff, attacked
406 
407     else:
408         print("Error en la introducción de la opción iterativa")
409 
410     return
411 
412 # Identificamos el nombre del archivo a importar
413 path = "data_sp.csv"
414 
415 # Creamos un dataframe a partir del csv y almacenamos el original
416 df = pd.read_csv(path, index_col="Unnamed: 0")
417 df_original = pd.read_csv(path, index_col="Unnamed: 0")
418 
419 ataques = ["DN", "DI", "BN", "BI", "CN", "CI", "EN", "EI"]
420 # ataques = ["DN"]
421 
422 df_gcc_abs = pd.DataFrame()
423 df_sls_abs = pd.DataFrame()
424 df_eff_abs = pd.DataFrame()
425 df_gcc = pd.DataFrame()
426 df_sls = pd.DataFrame()
427 df_eff = pd.DataFrame()
428 df_Rgcc = pd.DataFrame()
429 df_Rsls = pd.DataFrame()
430 df_Reff = pd.DataFrame()
431 df_attacked = pd.DataFrame()
432 
433 for i in range(len(ataques)):
434     ataque = ataques[i]
435     print(ataque)
436     if ataque[0] == "D":
437         attack = "degree"
438     elif ataque[0] == "B":
```

```
439             attack = "betweenness"
440         elif ataque[0] == "C":
441             attack = "closeness"
442         elif ataque[0] == "E":
443             attack = "eigenvector"
444 
445         if ataque[1] == "N":
446             iterativo = "NO"
447         elif ataque[1] == "I":
448             iterativo = "SI"
449 
450         # Llamamos a la función "final"
451         gcc, sls, eff, attacked = funcfinal(path, df, attack, iterativo)
452         Rgcc, gcc_rel = get_R_gcc(gcc)
453         Rsls, sls_rel = get_R_sls(sls)
454         Reff, eff_rel = get_R_eff(eff)
455         df_gcc_abs[ataques[i]] = pd.Series(gcc)
456         df_sls_abs[ataques[i]] = pd.Series(sls)
457         df_eff_abs[ataques[i]] = pd.Series(eff)
458         df_gcc[ataques[i]] = pd.Series(gcc_rel)
459         df_sls[ataques[i]] = pd.Series(sls_rel)
460         df_eff[ataques[i]] = pd.Series(eff_rel)
461         df_Rgcc[ataques[i]] = pd.Series(Rgcc)
462         df_Rsls[ataques[i]] = pd.Series(Rsls)
463         df_Reff[ataques[i]] = pd.Series(Reff)
464         df_attacked[ataques[i]] = pd.Series(attacked)
465 
466         # Volvemos a definir el dataframe original ya que lo hemos modificado
467         df = df_original.copy()
468 
469 # Guardamos en formato csv los dataframes que contienen los resultados
470 w = path[5:7]
471 df_gcc_abs.to_csv("Resultados_" + w + "/gccabs_ " + w + ".csv")
472 df_sls_abs.to_csv("Resultados_" + w + "/slsabs_" + w + ".csv")
473 df_eff_abs.to_csv("Resultados_" + w + "/effabs_" + w + ".csv")
```

```
474 df_gcc.to_csv("Resultados_" + w + "/gcc_" + w + ".csv")
475 df_sls.to_csv("Resultados_" + w + "/sls_" + w + ".csv")
476 df_eff.to_csv("Resultados_" + w + "/eff_" + w + ".csv")
477 df_Rgcc.to_csv("Resultados_" + w + "/Rgcc_" + w + ".csv")
478 df_Rsls.to_csv("Resultados_" + w + "/Rsls_" + w + ".csv")
479 df_Reff.to_csv("Resultados_" + w + "/Reff_" + w + ".csv")
480 df_attacked.to_csv("Resultados_" + w + "/attacked_" + w + ".csv")
481
482 # Creamos el grafo original
483 G = creategraph(df, path)
484
485 # Imprimimos la caracterización topológica de la red
486 print("G tiene", nx.number_of_nodes(G), "nodos y", nx.number_of_edges(G), "links")
487 print("La densidad de G es", np.round(nx.density(G),4))
488 print("El grado medio de G es", np.round(nx.number_of_edges(G)/nx.number_of_nodes(G),4))
489 print("El diámetro de G es", nx.diameter(G))
490 print("La longitud promedio de G es", np.round(nx.average_shortest_path_length(G),4))
491 print("El coeficiente medio de clustering de G es", np.round(nx.average_clustering(G),4))
492 print("La la eficiencia global media de G es", np.round(nx.global_efficiency(G),4))
```