Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Power domination sets and centralities in the European airport network

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: Sergio Fernández Bertolín

DIRECTOR: Francesc Comellas Padró

DATA: 1 de setembre del 2021

Títol: Power domination sets and centralities in the European airport network

Autor: Sergio Fernández Bertolín

Director: Francesc Comellas Padró

Data: 1 de setembre del 2021

Resum

Un conjunt dominador en potència (*power dominating set*) en una xarxa és un subconjunt de nodes que poden monitoritzar la totalitat de la xarxa a partir d'un conjunt de regles senzilles. El concepte s'introduí en el context de les xarxes elèctriques, però s'ha estès i estudiat per a altres xarxes i famílies de grafs.

S'ha demostrat que aquest problema és NP-complet pel que, per xarxes grans, mètodes d'optimització com la recuita simulada (*simulated annealing*), *threshold accepting* o algorismes genètics constitueixen eines útils per poder trobar solucions òptimes o quasi-òptimes.

En aquest TFG es presenta un nou algorisme de tipus llindar d'acceptació (*threshold accepting)* per trobar conjunts dominadors en potència a grafs i l'apliquem per a analitzar aquests conjunts en funció de les propietats de centralitat dels aeroports i possibles fallades en cascada en la xarxa Europea d'aeroports.

Primerament, aquesta xarxa s'analitza considerant diferents centralitats dels nodes, com la centralitat clàssica de grau, la d'intermediació, PageRank, etc. i així es poden identificar els nodes més connectats i tenir una visió global de la xarxa. També es dona, mitjançat la cerca de comunitats, una aproximació alternativa de la connectivitat local entre nodes. Aquests grups de nodes s´identifiquen en base a unes associacions locals fortes, en contraposició a l'associació més feble amb la resta de nodes.

Una vegada caracteritzada la xarxa des d'aquestes perspectives, es modelen les fallades en cascada per a detectar vulnerabilitats a la xarxa tot comparant els efectes dels tancament de diferents aeroports. La comparació es basa en la fallada dels aeroports en funció de la seva classificació. Els aeroports caiguts poden ser d'un conjunt dominador o corresponents a nodes d'alt, mitjà o baix grau de connectivitat.

El resultat principal és una visió completa de com el sistema queda afectat per la caiguda de nodes en funció de les seves propietats i de com els paràmetres utilitzats per a definir el model impacten a l'anàlisi final.

Título: Power domination sets and centralities in the European airport network

Author: Sergio Fernández Bertolín

Director: Francesc Comellas Padró

Date: 1st september 2021

Overview

A power domination set in a network is a subset of its nodes such that by applying a simple set of rules all nodes are monitored. The concept was introduced in the context of electrical networks but has been extended and studied for other networks and families of graphs.

It has been shown that this problem is NP-complete and thus, for large networks, optimization methods like simulated annealing, threshold acceptance or genetic algorithms could be useful to find near-optimal solutions.

In this TFG we introduce a new threshold acceptance algorithm to find power domination sets in graphs and we apply it to analyze these sets in relation to airport centralities and cascade failures in the European airport network.

To start with, the cited network is analyzed using a few distinct centrality definitions, including a classical degree centrality, betweenness centrality PageRank, etc. to find the most connected nodes and have an overview of the whole network. An alternative outlook of the local connectivity between nodes is given by finding communities. These groups of nodes are identified according to their strong local associations, in opposition to weaker associations with the rest of the nodes.

Having characterised the network from different perspectives, cascade failures are modelled to detect vulnerabilities on the network by comparing the effects of different airport closures. A comparison is assessed based on the effect of the failure of airports depending on their classification. Failing nodes could belong to power dominations sets or be nodes selected from high, medium or low connectivity levels.

The main result is a complete insight on how the system is affected from the failure of nodes according to their properties and how the parameters used to tune the model affect this analysis.

Aquest treball no hauria estat possible sense
el suport incondicional dels meus pares i la meva dona.

També mereix un agraïment molt especial el meu director,
en Francesc Comelles, per les nombroses facilitats que m'ha donat
i la paciència que ha tingut durant aquest llarg procés.

I finalment vull donar gràcies a Déu per posar totes aquestes persones
al meu camí i encaixar totes les peces per a que el TFG hagi estat possible.

ÍNDEX

INTRODUCCIÓ

El modelatge de xarxes d'aeroports com a xarxes complexes no és cap novetat i ens pot ajudar a entendre el funcionament i les fallades d'aquestes infraestructures vitals per a les economies mundials.

Amb aquest modelatge es poden explorar propietats de tot tipus com connectivitat, saturació de la xarxa a diversos punts i localitzar vulnerabilitats de les mateixes. Episodis com el tancament de l'espai aeri al 2010 a conseqüència de l'erupció del volcà islandès Eyjafjallajökull són difícilment previsibles. No obstant, anticipar les conseqüències de la caiguda d'un aeroport és un exercici que es pot fer per a preparar mesures d'emergència i plans alternatius de manera preventiva.

En aquest treball ens centrem en l'anàlisi de la xarxa europea d'aeroports, construïda amb dades reals de vols del 2018. S'analitzen propietats de centralitat que ens indiquen, com a graf, la importància a nivell de connectivitat dels aeroports que componen el conjunt. Com a anàlisi complementària es determinen comunitats, que ajuden a descobrir associacions locals d'aeroports amb una relació més forta respecte a la resta.

L'objectiu final és trobar fortaleses i vulnerabilitats del conjunt simulant atacs selectius a aeroports concrets i veient com afecten el funcionament global de la xarxa. El que interessa és calcular l'abast de la caiguda d'un tipus específic d'aeroport, segons criteris de centralitat i de dominància en potència (*power domination*), que es defineixen amb detall als Capítols 1 i 2.

Així, una de les parts més innovadores d'aquest treball és aplicar el concepte de dominància en potència, prou estès a l'anàlisi de xarxes elèctriques, a altres tipus de xarxa per ajudar a localitzar de manera efectiva, en aquest cas, aeroports rellevants en relació a possibles fallades globals de la xarxa.

CAPÍTOL 1. GRAFS

1.1. Grafs

Un graf *G* es pot representar mitjançant un diagrama que mostra l'estructura d'una xarxa i com els diversos elements d'aquesta interactuen entre ells [1]. Per exemple, la interconnexió dels ordinadors d'un departament en una xarxa d'àrea local es pot representar en forma de graf.

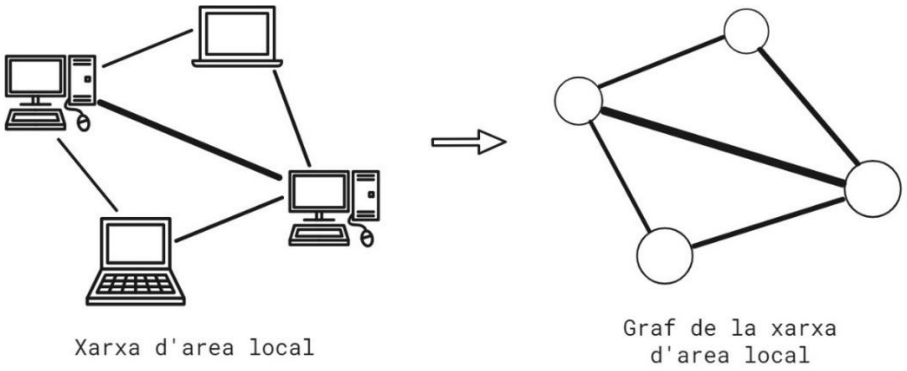

Fig. 1.1 Graf d'una xarxa d'àrea local

Matemàticament, un graf (simple, no dirigit) *G = (V, E),* és un parell ordenat constituït per un conjunt *V* de vèrtexs o nodes i un conjunt *E* d'arestes o enllaços, els quals són formats per parelles d'elements del conjunt *V*.

Per a estudiar un graf convé conèixer els seus components i paràmetres fonamentals. A continuació en llistem els més rellevants pel treball a presentar.

1.1.1. Paràmetres dels grafs

1.1.1.1. Nodes i arestes

Els components de la xarxa s'anomenen nodes i els enllaços per on interactuen són les arestes. Una xarxa té N nodes i L arestes. A l'exemple anterior els nodes serien els ordinadors i les arestes les línies de comunicació entre ells. El graf tindria N=4 nodes i L=5 arestes.

1.1.1.2. Grafs dirigits i no.dirigits

Les arestes poden connectar els nodes només en un sentit o en els dos sentits possibles. Si totes les arestes són dirigides el graf també ho serà i igualment un graf amb totes les arestes no dirigides serà no dirigit. En aquest treball només ens ocuparem de grafs no dirigits.

1.1.1.3. Grau d'un node

El grau d'un node correspon al nombre total d'arestes que té en comú amb altres nodes.

1.1.1.4. Grau mitjà d'un graf no dirigit

El grau mitjà d'un graf no dirigit és la mitjana dels graus de tots els nodes del graf. Es defineix així.

$$GM = \frac{1}{N}\sum_{i=1}^{N} k_i \qquad \textbf{(1.1)}$$

On N és el nombre de nodes del graf i $k_i$correspon al grau del node i.

1.1.1.5. Distribució de graus

La distribució de graus mostra la probabilitat que té qualsevol node del graf de tenir un grau determinat. La distribució de graus és un factor informatiu clau sobre la xarxa estudiada i és necessari per al càlcul d'altres propietats de la xarxa. El càlcul de la probabilitat p_k de que un node tingui grau k és una simple divisió.

$$p_k = \frac{N_k}{N} \qquad \textbf{(1.2)}$$

N_k és el nombre de nodes amb grau k i N el nombre total de nodes al graf. La figura següent mostra una xarxa molt simple i la distribució de graus dels seus nodes.

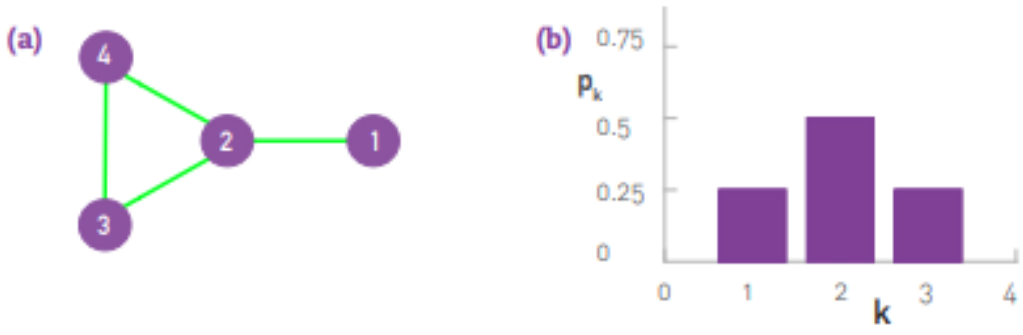

Fig. 1.2 Graf simple (a) amb la seva distribució de graus (b) [1]

1.1.1.6. Camí més curt

La seqüencia de nodes que connecta de manera consecutiva diferents nodes recorrent les serves arestes és un camí. Entre dos nodes qualsevol del graf hi pot haver diversos camins. El camí que recorre menys arestes correspon al camí més curt.

1.1.1.7. Distància entre nodes

Anomenem distància entre nodes a la distància definida pel camí més curt entre els dos nodes estudiats. En el cas de grafs no dirigits, la distància del node **i** al node **j** és la mateixa que la del node **j** al node **i**.

1.1.1.8. Diàmetre de la xarxa

El diàmetre de la xarxa correspon al major camí més curt de tots els que es poden definir entre tots els parells de nodes de la xarxa. Per exemple, al graf de la figura 1.2 el diàmetre és 2, que correspon a la distància mínima de camí més curt amb la que podem connectar qualsevol parell de nodes.

1.1.1.9. Pesos d'una aresta

Si no s'indica el contrari es considera que totes les arestes tenen pes unitari. No obstant, modelar les arestes amb pesos diferents ajuda a caracteritzar la xarxa segons les capacitats o càrregues de cada aresta. Assignem llavors un pes w_{ij} diferent a cada aresta entre els nodes **i**, **j** per a modelar aquesta característica. El repte més complicat en ponderar els pesos és que aquesta assignació representi de manera acurada la característica a visualitzar de la xarxa.

1.1.1.10. Coeficient d'agrupament

El coeficient d'agrupament (*Clustering Coefficient*) defineix fins a quin punt els veïns d'un node estan connectats entre ells. El coeficient d'agrupament, CA_i del node **i** es computa amb la següent equació.

$$CA_i = \frac{2L_i}{k_i(k_i-1)} \qquad \textbf{(1.1)}$$

Amb L_i el nombre d'arestes entre els veïns del node **i**. k_i és el grau del node **i**.

CAPÍTOL 2. PARÀMETRES DE XARXES

2.1 Centralitats

Les centralitats són mesures de la xarxa que ens permeten quantificar la importància dels nodes a la xarxa segons diversos criteris basats en la connectivitat amb altres nodes.

2.1.1. Centralitat de grau

La centralitat de grau (*Degree Centrality*) és la mesura més simple que estima la connectivitat a la xarxa i es calcula comptant el nombre d'arestes que un node comparteix amb altres, és a dir, el seu grau [2]. Es defineix així:

$$C_D(i) = \sum_{j=1}^{n} a_{ij} \qquad \textbf{(2.1)}$$

On **i** correspon al node calculat i a_{ij} és 1 quan existeix una aresta entre els nodes **i**, **j**. En cas de no haver cap aresta entre els nodes, a_{ij} val 0. Si el graf és dirigit calculem una centralitat de grau per a les arestes entrants i una altra diferent per a les arestes sortints.

2.1.2. Centralitat de proximitat

La centralitat de proximitat (*Closeness Centrality*) indica en quin grau un node és proper respecte a la resta de nodes que formen la xarxa avaluant la seva distància mínima [3].

$$C_C(i) = \frac{n-1}{\sum_{j \neq i} d_{ij}} \qquad \textbf{(2.2)}$$

On n és el nombre de nodes i d_{ij} correspon a la distància del camí més curt entre els nodes **i**, **j**. Així un valor proper a n voldrà dir que el node té camins molt curts a la resta de nodes i un valor proper a zero voldria dir que el node té camins molt llargs respecte a la resta de nodes.

2.1.3. Centralitat d'intermediació

La centralitat d'intermediació (*Betweenness Centrality*) quantifica fins a quin punt un node es troba al camí dels altres nodes [2]. El càlcul és un quocient entre el nombre de camins més curts que passen per un node respecte al nombre total de camins mínims.

$$C_B(i) = \sum_{k \neq i \neq j \in N} \frac{\sigma_{kj}(i)}{\sigma_{kj}} \quad \textbf{(2.3)}$$

On $\sigma_{kj}(i)$ és el nombre de camins més curts entre els nodes **k**, **j** que passen pel node **i**. El denominador σ_{kj} és el nombre total de camins més curts entre el nodes **k**, **j**. És així com el valor màxim d'aquest índex pot ser 1 i indicaria que tots els camins més curts passen pel node en qüestió.

2.1.4. Centralitat de comunicabilitat d'intermediació

La centralitat de comunicabilitat d'intermediació (*Communicability Betweenness Centrality*) mesura la centralitat de cada node en base a totes les rutes entre cada parell de nodes del graf i les que discorren pel node a estudiar. [4]

$$C_{BC}(i) = \frac{1}{C}\sum_j \sum_k \frac{G_{ijk}}{G_{jk}},\ j \neq k,\ j \neq i, k \neq i \quad \textbf{(2.4)}$$

G_{ijk} és la suma ponderada de tots els camins que inclouen el node **i**. Per finalitzar, G_{jk} correspon al nombre total de camins entre tots els parells de nodes. C correspon al nombre total de termes del sumatori i és igual a $(n-1)^2 - (n-1)$. Aquest terme correspon a un factor de normalització que acota la centralitat exclusivament a valors entre 0 i 1.

2.1.5. Centralitat de proximitat de flux de corrent

La centralitat de proximitat de flux de corrent (*Current Flow Betweenness Centrality* o *Random-walk Betweenness*) és una altra mesura de la rellevància dels nodes, en aquest cas fent servir propietats físiques, tal com es descriu a l'article de Newman [5]. Es converteix l'estructura del nostre graf en un circuit elèctric considerant resistències unitàries a cada aresta i que a cada node entrant s'aplica una unitat de corrent i una unitat de corrent a cada node sortint. A la figura següent, extreta de l'article de Newman [5], en tenim un exemple visual.

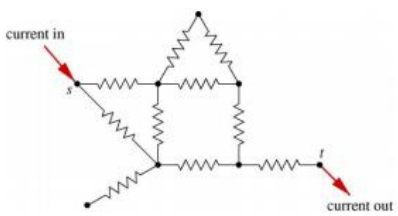

Fig. 2.1 Circuit elèctric resultant d'un graf per al càlcul de la centralitat de proximitat de flux de corrent

Amb les lleis de Kirchhoff per estudiar corrents i voltatges podem mesurar de forma matricial l'aportació de corrent a cada node.

$$C_{Fl}(i) = \frac{\sum_{s<t} I_i^{(st)}}{\left(\frac{1}{2}\right)n(n-1)} \qquad \textbf{(2.5)}$$

On el numerador correspon a la suma d'intensitats de corrents al node estudiat i la n del denominador és el nombre total de nodes.

2.1.6. Centralitat PageRank

La centralitat PageRank (*PageRank Centrality*) és la centralitat que fan servir cercadors web com Google. Es modelen les pàgines web com a nodes i els hipervincles entre elles com a arestes. Una pàgina té un PageRank alt si és apuntada per moltes pàgines, que a la vegada són apuntades per moltes altres pàgines.

Més exactament, la centralitat PageRank de cada node ve donada per l'autovector de la matriu $\boldsymbol{C_{PR}}$ corresponent a l'autovalor més gran. Aquesta matriu nxn es defineix com

$$\boldsymbol{C_{PR}} = \propto \boldsymbol{S} + \frac{1-\propto}{n}\boldsymbol{J} \qquad \textbf{(2.6)}$$

On n és el nombre de nodes del graf, $\propto$ és un paràmetre d'ajust (típicament 0.85), $\boldsymbol{J}$ la matriu on totes les entrades són 1 i $\boldsymbol{S}$ és una certa matriu estocàstica obtinguda a partir de la matriu d'adjacència del graf que dóna les connexions entre el nodes de la xarxa [6].

2.1.7. Comunitats

Una altra propietat a destacar de les xarxes és l'agrupament de nodes en estructures anomenades comunitats [7]. Aquestes estan formades pels conglomerats de nodes que tenen una densitat d'arestes entre ells superior a les arestes amb nodes d'altres comunitats. Dit d'una altra manera, els nodes de cada comunitat estan molt ben connectats entre ells i la connexió amb els nodes de comunitats externes és molt inferior. Aquest nivell de connexió es determina pels pesos de les arestes.

2.2 Conjunt dominador de potència

El conjunt dominador de potència (*Power Dominating Set*) és un subconjunt de nodes d'un graf, que compleix uns requeriments molt específics que permeten considerar que aquest conjunt abasta tota la xarxa. El concepte prové originàriament de les xarxes elèctriques i s'aplicà per trobar els millors nodes per a instal·lar dispositius de monitoratge i control de la xarxa [8].

Un conjunt dominador de potència ha de ser capaç d'observar la xarxa sencera. Per garantir aquesta observació completa considerem que un node és observat si compleix les següents condicions.

1. Totes les arestes incidents als nodes del conjunt inicial són arestes observades.
2. Qualsevol node incident a les arestes observades es considera observat.
3. Les arestes que enllacin dos nodes observats són arestes observades al mateix temps.
4. Si un node és incident a un número d'arestes k>1 i totes són observades menys una, llavors totes les arestes del node són observades

En base a aquestes regles, per a obtenir el total de nodes observats utilitzarem el següent procediment, on P és el conjunt de nodes candidat a dominador en potència.

1. Assignem d'inici el conjunt C=P i F el conjunt d'arestes incidents a P.
2. Afegim al conjunt C tots els vèrtexs que no pertanyin al conjunt i siguin incidents a una aresta del conjunt F.
3. Incorporem al conjunt F qualsevol aresta que encara no sigui del conjunt i compleixi alguna d'aquestes dues condicions:
    a. Enllaci dos nodes del conjunt C.
    b. Sigui incident a un node v amb totes les seves arestes incidents a C menys una, sempre i quan el node v sigui de grau superior a 1.
4. Si no hem afegit cap nova aresta o node en els passos 2 i 3, acabem aquí. En cas contrari, tornem al pas 2.

2.3 Algorismes d'optimització

2.3.1 Recuita simulada

La recuita simulada o *Simulated Annealing (SA)* és un algorisme d'optimització estocàstic [9]. Mitjançant aquest algorisme es destrien solucions que optimitzen el problema, segons una certa funció de qualitat. Per fer-ho, partim d'una solució inicial que anem refinant provant diverses configuracions de manera iterativa, cadascuna similar a l'anterior. Si la nova solució millora la funció de qualitat definida, s'accepta com a nova solució millor. Si no, és acceptada amb una probabilitat que va decreixent amb el pas de les iteracions. Aquesta probabilitat depèn d'un paràmetre anomenat temperatura (**T**). D'aquesta manera podem explorar algunes solucions que, tot i empitjorar el resultat, estiguin més properes a una altra solució millor que les anteriors.

El pseudo-codi de l'algorisme seria el següent:

1. Escollir paràmetres inicials $\mathbf{S_0}$ (solució inicial) i **T** (temperatura)

2. Generar una nova solució $\mathbf{S_1}$ amb un petit canvi respecte a l'anterior i avaluar l'increment en la funció de qualitat (**ΔQ**):

$$\mathbf{\Delta Q} = \text{Qualitat}(\mathbf{S_1}) - \text{Qualitat}(\mathbf{S_0}) \qquad \mathbf{(1.1)}$$

3. Si **ΔQ**>0, llavors acceptar la nova solució, $\mathbf{S_0} = \mathbf{S_1}$
   En cas contrari, acceptar la nova solució amb probabilitat exp(**ΔQ** /**T**)

4. Si transcorre un nombre d'iteracions **MAX_ITER** sense cap millora, llavors decrementar el paràmetre **T**. Típicament **T**=**T***0.99.

5. Si transcorre un temps prefixat sense canvis finalitza l'algorisme. En cas contrari tornar al punt 2.

2.3.2 *Threshold Accepting*

El *Threshold Accepting (TA)*, és un altre algorisme d'optimització estocàstic basat en la recuita simulada i molt similar al mateix [9]. El principal canvi és el llindar d'acceptació utilitzat, que ara és directament la temperatura T. A més, ara acceptem qualsevol solució que estigui dintre del llindar en comptes d'acceptar-les amb una probabilitat. El pseudo-codi de l'algorisme és el següent:

1. Escollir paràmetres inicials $\mathbf{S_0}$ (solució inicial) i **T** (temperatura)

2. Generar una nova solució $\mathbf{S_1}$ amb un petit canvi respecte a l'anterior i avaluar l'increment en la funció de qualitat (**ΔQ**):

$$\mathbf{\Delta Q} = \text{Qualitat}(\mathbf{S_1}) - \text{Qualitat}(\mathbf{S_0}) \qquad \mathbf{(1.2)}$$

3. Si **ΔQ**<T, llavors acceptar la nova solució, $\mathbf{S_0} = \mathbf{S_1}$

4. Si transcorren un nombre d'iteracions **MAX_ITER** sense cap millora, decrementar el paràmetre **T**. Típicament **T**=**T***0.99.

5. Si transcorre un nombre d'iteracions prefixat sense canvis finalitza l'algorisme. En cas contrari tornar al punt 2.

Aquesta petita modificació de l'algorisme el fa més simple d'aplicar i ajuda a obtenir resultats millors en temps més reduïts, com s'indica a l'article de Dueck [9].

CAPÍTOL 3. FALLADES EN CASCADA

3.1 Fallades en cascada

La robustesa d'una xarxa es pot posar a prova de diverses maneres, una d'elles és simular la fallada de nodes aleatoris. De més interès per a descobrir vulnerabilitats estructurals és la fallada en cascada, on els nodes inicials que fallen no són aleatoris sinó els que compleixen uns requisits determinats [10] i transmeten la fallada a nodes veïns.

Diversos estudis consideren que les fallades locals redistribueixen la càrrega als nodes veïns, podent provocar la caiguda de la xarxa sencera per col·lapse quan els nodes que fallen inicialment són uns nodes específics [11, 12].

En diversos treballs es compara el resultat de la fallada en cascada quan el conjunt de nodes atacats (o que fallen inicialment) tenen una càrrega gran, petita o mitjana, però aquesta càrrega es pot definir de diverses maneres.

3.1.1 Definició de càrrega

En un gran nombre d'estudis la càrrega assignada a un node es decideix en base al seu grau o a la seva centralitat d'intermediació, tal i com s'ha definit al capítol anterior i com tornem a presentar a l'equació (3.1).

$$C_{BC}(i) = \frac{1}{C}\sum_j \sum_k \frac{G_{ijk}}{G_{jk}},\ j \neq k,\ j\ \neq i, k \neq i \qquad \textbf{(3.1)}$$

Aquestes dues aproximacions tenen els seus inconvenients. La primera, basada en el grau, és molt simple d'aplicar però obvia gran quantitat d'informació de la xarxa a estudiar. La segona té en compte un nombre de característiques més ampli però es converteix en un còmput molt costós en xarxes de dimensió considerable.

És així com al treball de referència d'aquest document [10], es presenta una nova mesura de càrrega que balanceja la quantitat d'informació emprada amb la computació de la mesura. Per fer-ho, s'assumeix que la càrrega inicial del node **j**, L_j es basa en el grau del propi node, k_j i els dels veïns, k_m amb la relació següent.

$$L_j = \left[k_j(\sum_{m \in \Gamma_j} k_m)\right]^{\propto} \qquad \textbf{(3.2)}$$

$\propto$ correspon a un paràmetre ajustable de 0 a 1 que controla la influència de la càrrega inicial dels nodes, incrementant aquest a mida que s'acosta al seu valor màxim 1.

3.1.2 Redistribució de càrrega

Quan un node és atacat i cau la seva càrrega es redistribueix als nodes veïns de manera proporcional en funció a les càrregues L_i del node **i** en fallada, la càrrega L_j del node **j** a on desviem la càrrega i les càrregues L_n. dels veïns del node atacat. Aquest redistribució es modela amb l'equació (3.3).

$$\Delta L_{ji} = L_i \frac{L_j}{\sum_{n \in \Gamma_i} L_n} \qquad \textbf{(3.3)}$$

On ΔL_{ji} és l'augment de càrrega al node **j** provinent del node caigut **i**. Γ_i és el conjunt de veïns del node **i**. Wang i Rong [10], mostren de manera molt gràfica aquesta redistribució amb la figura 3.1.

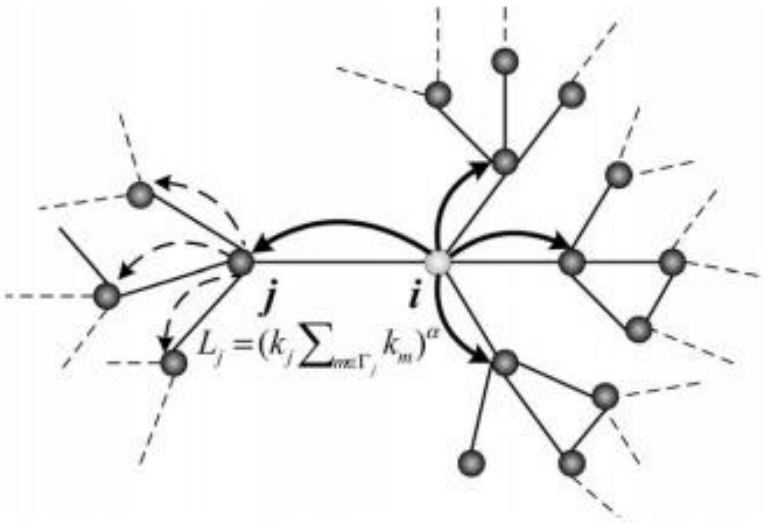

Fig. 3.1 Redistribució de càrrega del node **i** col·lapsat al seu veïnatge [10]

3.1.3 Capacitat del node i nodes caiguts

Amb l'increment de càrrega produït per la fallada dels nodes seleccionats, la següent pregunta a fer seria quin és l'increment màxim de càrrega que es pot absorbir a la xarxa. Novament al mateix treball [10], se suggereix que cada node

té una capacitat C proporcional a la seva càrrega L, de manera que modelem la càrrega C_i del node **i** com segueix.

$$C_i = TL_i \quad \textbf{(3.5)}$$

On T és un factor de tolerància superior a 1. Si la càrrega suportada per cada node, que correspon a la pròpia més la incrementada pel desviament des dels nodes amb fallada, $L_j + \Delta L_j$ és superior a la càrrega, $C_j = TL_j$ el node es considera caigut.

$$L_j + \Delta L_j > TL_j \quad \textbf{(3.6)}$$

D'aquest manera cada node caigut genera una nova distribució de càrrega cap als altres, podent generar al seu torn noves caigudes. Amb totes les caigudes en cascada calculades es quantifiquen els nodes que han caigut.

Denotem CF_i com la mida d'allau del node **i**, que indica el nombre de nodes que cauen després de la fallada del node **i**. CF_i tindrà valors entre 0 i N-1, on N és el nombre total de nodes de la xarxa.

Finalment, per quantificar la robustesa total de la xarxa, s'utilitza la mida d'allau normalitzada.

$$CF_{attack} = \frac{\sum_{i \in A} CF_i}{N_A(N-1)} \quad \textbf{(3.7)}$$

Aquest paràmetre ens donarà valors entre 0 i 1. El valor 0 indica integritat total de la xarxa i 1 col·lapse total de la mateixa. $N_A$és el nombre de nodes atacats i A el conjunt d'aquests. CF_i, tal com s'ha presentat al paràgraf anterior, és el nombre de nodes caiguts per l'atac del node **i**.

CAPÍTOL 4. XARXA EUROPEA D'AEROPORTS

4.1 Característiques de la xarxa

L'objecte d'estudi d'aquest treball és la xarxa europea d'aeroports amb informació dels vols entre inicis de maig de 2017 i finals d'abril de 2018. Aquesta ha estat modelada per Gelabert [13] amb dades obtingudes de Datahub.io, d'Eurocontrol i d'Opensky-Network.

Per simplificar l'estudi de la mateixa s'ha assumit que totes les rutes són simètriques (pràcticament tots els vols fan trajectes d'anada i tornada i la quantitat de vols que no ho fan és negligible comparativament). És així com s'arriba a una xarxa no dirigida on les arestes tenen com a pes el nombre d'operacions totals entre nodes en l'any d'observació, independentment del sentit de desplaçament.

Una primera inspecció visual, a la figura 4.1, ens permet identificar els enllaços i el total d'operacions de cada aeroport, que es correspon amb la mida del node.

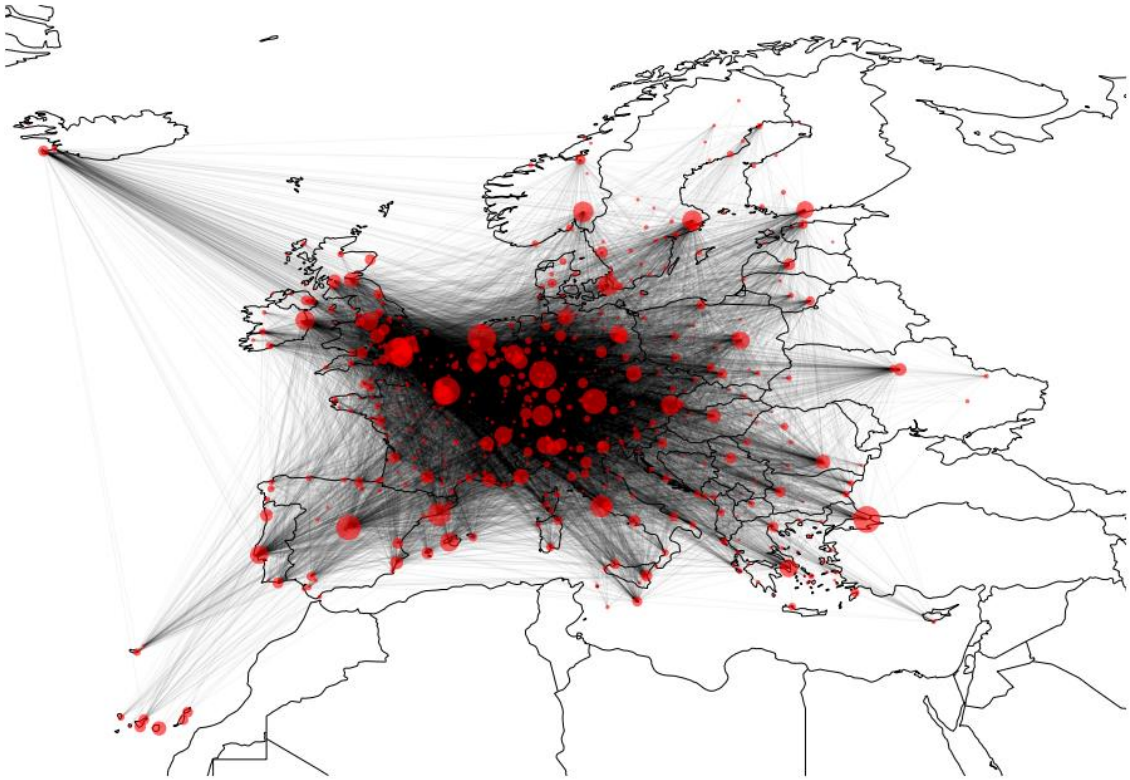

Fig. 4.1 Xarxa europea d'aeroports

Per a no perdre les propietats i connectivitats més rellevants, estudiarem aquesta xarxa i una altra versió simplificada que elimina enllaços amb un pes inferior a 17. És a dir, les rutes amb menys de 17 vols en l'any d'estudi s'obvien a la xarxa simplificada. De la mateixa manera, els nodes que quedin desconnectats i no tinguin cap aresta després d'aquesta eliminació també s'han de treure del nou graf. Es pretèn amb aquesta decisió poder eliminar efectes de soroll al global de la xarxa. L'elecció de 17 vols com a punt de tall per a la simplificació no és arbitrària, ja que ens quedem amb tots els destins que tenen un nombre de vols anuals mínimament representatiu (almenys un cada tres setmanes, en promig). També es redueix el grau mitjà de la xarxa de manera considerable, de 90 nodes als poc més de 24 de la xarxa reduïda.

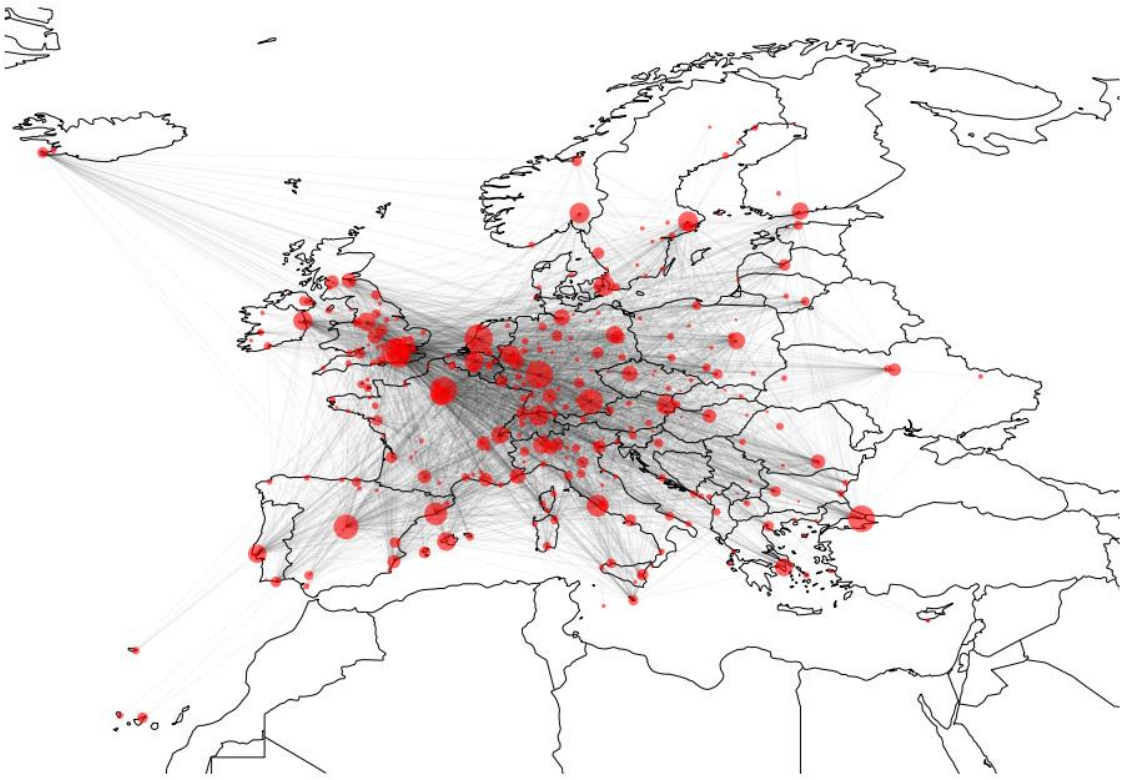

Fig. 4.2 Xarxa europea d'aeroports simplificada

A la figura 4.2 veiem clarament que hi ha menys arestes i també la desaparició d'alguns nodes, per exemple la meitat dels aeroports de les Illes Canàries o d'altres zones perifèriques com al Nord del Mar Bàltic.

Passem de la simple observació a la quantificació estudiant els principals paràmetres d'aquestes dues xarxes, que s'han definit al capítol 1 i es llisten a la taula 4.1.

Taula 4.1. Paràmetres de la xarxa completa i la simplificada

Paràmetre	Xarxa completa	Xarxa simplificada
Nombre de nodes	509	345
Nombre d'arestes	22789	4184
Grau mitjà	89,54	24,25
Grau màxim	331	150
Diàmetre de xarxa	5	6
Distància mitjana entre nodes	1,96	2,41
Coeficient d'agrupació mitjà	0,63	0,51

Amb la simplificació, el nombre d'arestes ha quedat pràcticament en un setena part de la mida original i el grau es redueïx a més d'una tercera part, pel que la nova xarxa mostra una complexitat molt menor i una estructura que serà més equilibrada i menys dependent dels nodes grans. Visualitzem a l'histograma de la figura 4.3 com a la xarxa original existeixen diversos nodes de grau superior a 300 i com les probabilitats més altes es concentren en graus petits.

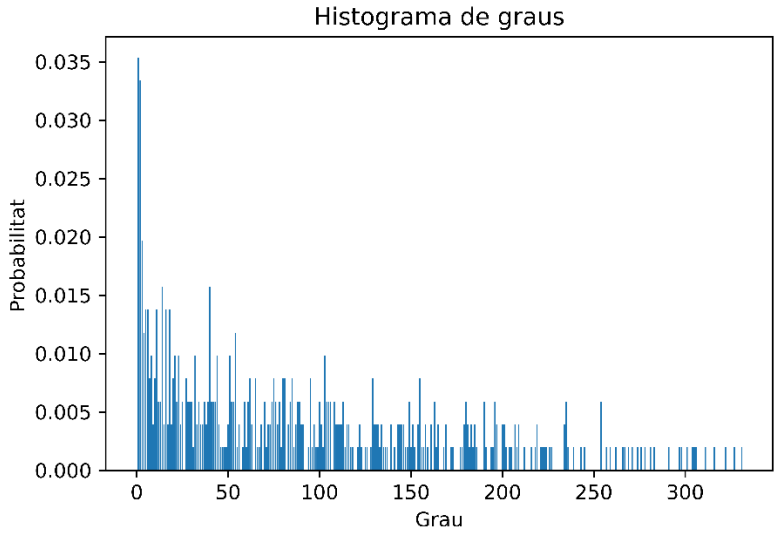

Fig. 4.3 Distribució de graus del graf complet

La nova xarxa sense arestes de pes inferior a 17 (veure figura 4.4), per contra, té un grau màxim de 150 i encara més concentració de graus a valors inferiors, el que demostra la simplificació clara de connexions.

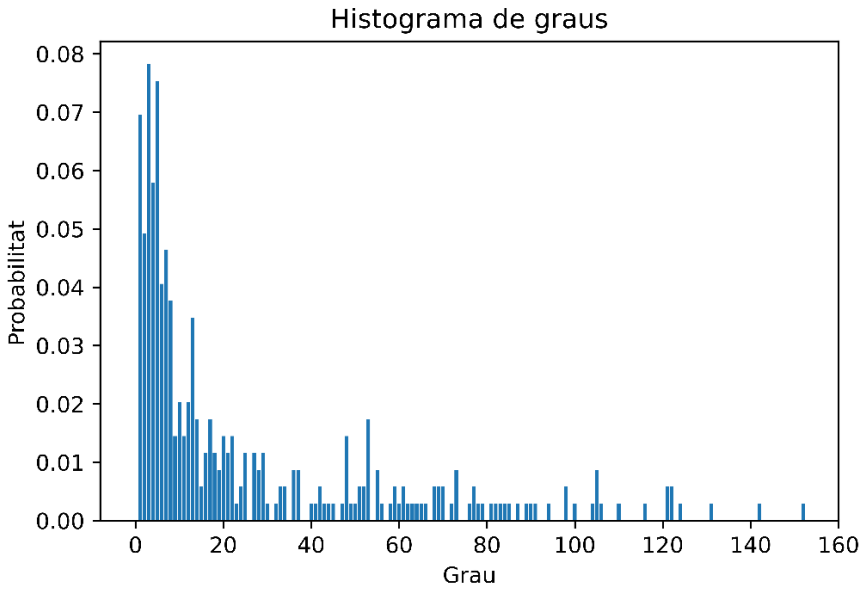

Fig. 4.4 Distribució de graus del graf simplificat

4.2 Comunitats de la xarxa

Al capítol 2, apartat 2.1.7, s'ha introduït el concepte de comunitat, que agrupa els nodes més connectats entre sí i menys connectats amb la resta de la xarxa en conjunts anomenats comunitats.

Fem servir l'algorisme introduït per Clauset i Newman a [7] per a destacar de manera molt visual la organització de la xarxa d'aeroports completa i què canvia en simplificar-la. Presentem el mapa amb les 4 comunitats trobades per a la xarxa sencera (figura 4.5), on veiem que hi ha una divisió marcada entre aeroports del NE (nodes vermells) i del SW (nodes blaus).

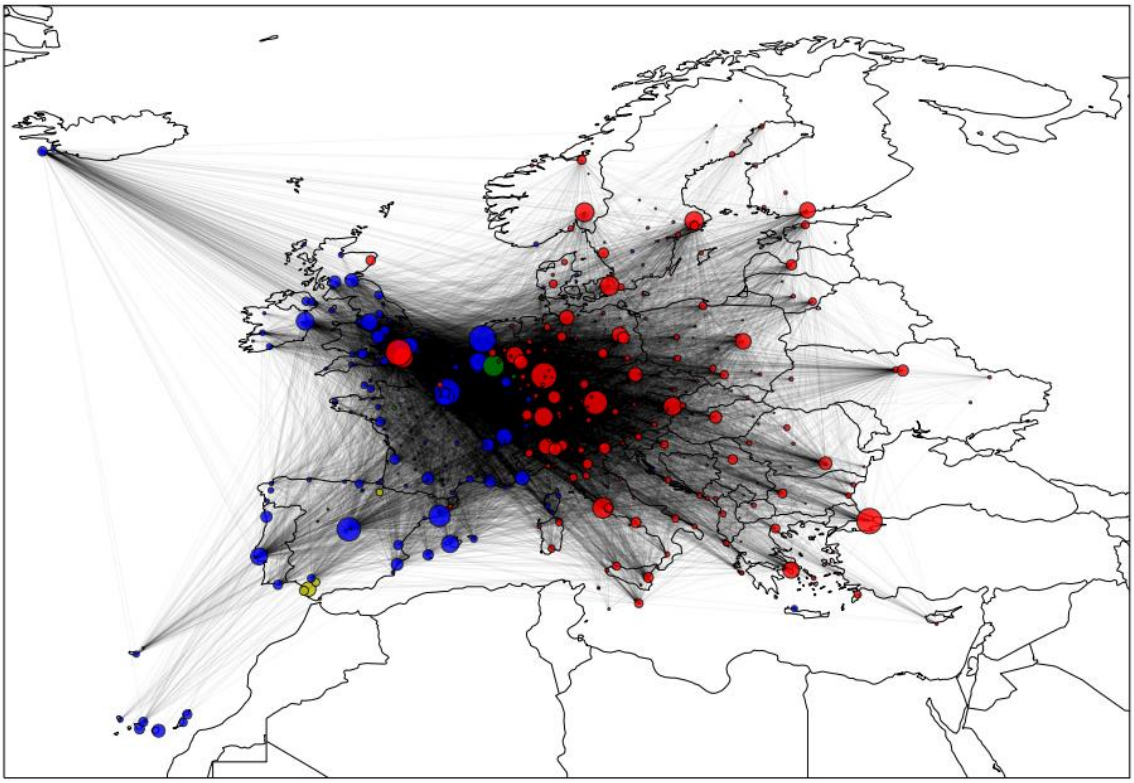

Fig. 4.5 Comunitats de la xarxa completa

Els cercles a cada node són proporcionals al nombre anual de vols. S'aprecien dues comunitats més petites, una amb 4 nodes grocs a Espanya, tres d'ells a Andalusia i una altra comunitat al Nord de França de color verd. Aquests nodes representen el nombre de vols a una escala 10 vegades superior a la de la resta de comunitats, ja que d'altra manera no serien visibles al mapa. Això ens dóna una idea de la relativa poca importància d'aquests aeroports i comunitats.

Explorem també les comunitats trobades a la xarxa simplificada, on trobem 3 comunitats grans i una petita. Per a visualitzar els nodes de la petita, de color groc i situats principalment a Noruega a la figura 4.6, hem hagut de multiplicar per 10 la mida proporcional al nombre de vols.

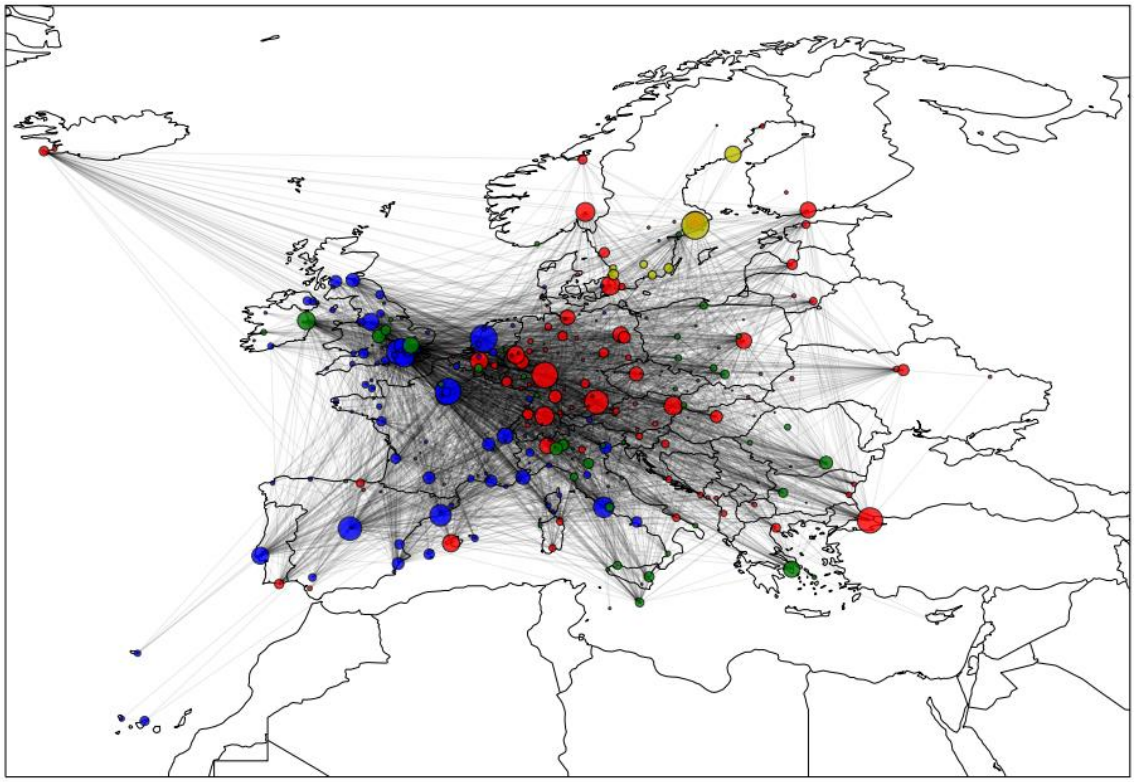

Fig. 4.6 Comunitats de la xarxa simplificada

Com es veu a la figura anterior, no només queda una única comunitat poc rellevant, sinó que ara en tenim 3 de significatives en front de les dues que teníem al graf complet. Una comunitat es concentra al NE i centre d'Europa (nodes vermells), una segona amb nodes blaus al SW i una tercera amb diversos nodes a una línia cap al centre d'Europa de color verd.

Per a aprofundir en el coneixement d'aquestes comunitats farem una anàlisi de centralitats, definides també al capítol 2, destacant els nodes de cada comunitat amb majors centralitats.

4.3 Centralitats de la xarxa

Iniciem l'estudi amb la taula 4.2. Hi destaquem, per a cada comunitat de la xarxa completa, els 5 nodes amb major coeficient de centralitat d'intermediació. Als casos de les comunitats 1 i 3, on només hi ha 4 i 3 nodes respectivament, és que aquella comunitat no té cap més integrant.

Taula 4.2. Centralitats de la xarxa completa per comunitat

Comunitat	Aeroport	Centralitat d'intermediació		Nombre d'operacions		Coeficient d'agrupament	Centralitat de grau	Centralitat de comunicabilitat d'intermediació		Centralitat de proximitat de flux de corrent		Centralitat de rang de pàgina	
		Total	Total/màxim	Total	Total/màxim			Total	Total/màxim	Total	Total/màxim	Total	Total/màxim
1	Jerez	0,0000	0,00	15142	0,03	0,82	0,07	0,05	0,06	0,02	0,80	0,0010	0,14
1	Moron Air Base	0,0000	0,00	4676	0,01	0,69	0,03	0,01	0,01	0,01	0,58	0,0006	0,09
1	Rota Naval Station	0,0000	0,00	5741	0,01	0,65	0,02	0,00	0,00	0,01	0,54	0,0005	0,08
1	Pamplona	0,0000	0,00	3534	0,01	0,77	0,05	0,02	0,03	0,02	0,74	0,0008	0,11
2	Paris-Le Bourget	0,0237	0,81	53955	0,11	0,34	0,65	0,80	**1,00**	0,02	**1,00**	0,0067	**1,00**
2	Amsterdam Schiphol	0,0231	0,79	508299	**1,00**	0,35	0,63	0,79	0,99	0,02	1,00	0,0067	1,00
2	Charles de Gaulle International	0,0204	0,70	482678	0,95	0,36	0,62	0,78	0,98	0,02	1,00	0,0065	0,97
2	London Stansted	0,0188	0,64	188156	0,37	0,35	0,61	0,77	0,96	0,02	1,00	0,0063	0,94
2	Brussels	0,0185	0,63	232719	0,46	0,37	0,60	0,77	0,96	0,02	1,00	0,0063	0,93
3	Liäge	0,0035	0,12	32483	0,06	0,55	0,35	0,51	0,64	0,02	0,97	0,0036	0,53
3	Spangdahlem Air Base	0,0000	0,00	1746	0,00	0,59	0,03	0,00	0,00	0,01	0,56	0,0005	0,08
3	Laval-Entrammes	0,0000	0,00	1175	0,00	**1,00**	0,01	0,00	0,00	0,01	0,28	0,0004	0,05
4	Eleftherios Venizelos International	0,0292	**1,00**	189681	0,37	0,48	0,36	0,51	0,63	0,02	0,97	0,0053	0,79
4	Stockholm-Arlanda	0,0283	0,97	248865	0,49	0,42	0,46	0,62	0,78	0,02	0,99	0,0057	0,84
4	Oslo Gardermoen	0,0211	0,72	251193	0,49	0,47	0,44	0,63	0,79	0,02	0,98	0,0053	0,79
4	Munich	0,0195	0,67	401847	0,79	0,38	0,60	0,78	0,97	0,02	1,00	0,0064	0,95
4	Zürich	0,0178	0,61	263549	0,52	0,35	0,64	0,80	1,00	0,02	1,00	0,0065	0,97

Apareixen dues comunitats, la 1 i la 3, amb nodes molt comunicats entre ells però amb una connectivitat a la xarxa global irrisòria, amb valors insignificants per a totes les centralitats, excepte per al coeficient d'agrupament on els valors són clarament més elevats. Les comunitats 2 i 4 contenen diversos aeroports amb coeficients considerables, concentrant valors màxims i propers al màxim per a cada tipus de centralitat. Comparem a la següent plana amb les comunitats de la xarxa simplificada per a veure la riquesa de l'aportació d'aquesta xarxa.

Taula 4.3. Centralitats de la xarxa simplificada per comunitat

Comunitat	Aeroport	Centralitat d'intermediació		Nombre d'operacions		Coeficient d'agrupament	Centralitat de grau	Centralitat de comunicabilitat d'intermediació		Centralitat de proximitat de flux de corrent		Centralitat de rang de pàgina	
		Total	Total/màxim	Total	Total/màxim			Total	Total/màxim	Total	Total/màxim	Total	Total/màxim
1	Stockholm-Arlanda	0,0496	0,55	248865	0,49	0,40	0,27	0,61	0,79	0,01	0,98	0,0112	0,65
1	Vienna International	0,0367	0,41	240095	0,47	0,37	0,30	0,67	0,86	0,01	0,99	0,0114	0,66
1	Munich	0,0352	0,39	401847	0,79	0,32	0,35	0,74	0,95	0,01	0,99	0,0128	0,75
1	Zürich	0,0338	0,38	263549	0,52	0,34	0,31	0,66	0,85	0,01	0,99	0,0116	0,68
1	Berlin-Schönefeld	0,0276	0,31	100121	0,20	0,37	0,30	0,66	0,85	0,01	0,99	0,0109	0,63
2	Amsterdam Schiphol	0,0693	0,77	508299	**1,00**	0,27	0,41	0,78	**1,00**	0,01	1,00	0,0159	0,93
2	London Luton	0,0584	0,65	135083	0,27	0,28	0,35	0,68	0,87	0,01	0,99	0,0139	0,81
2	London Gatwick	0,0409	0,46	285941	0,56	0,27	0,35	0,67	0,86	0,01	0,99	0,0131	0,76
2	Barcelona International	0,0400	0,45	323470	0,64	0,30	0,38	0,76	0,97	0,01	1,00	0,0136	0,79
2	Toulouse-Blagnac	0,0395	0,44	98991	0,19	0,41	0,18	0,36	0,46	0,01	0,96	0,0083	0,48
3	London Stansted	0,0898	**1,00**	188156	0,37	0,22	0,44	0,76	0,98	0,01	**1,00**	0,0172	**1,00**
3	Dublin	0,0334	0,37	222326	0,44	0,36	0,33	0,72	0,93	0,01	0,99	0,0120	0,70
3	Eleftherios Venizelos International	0,0241	0,27	189681	0,37	0,59	0,17	0,44	0,57	0,01	0,96	0,0072	0,42
3	East Midlands	0,0153	0,17	67490	0,13	0,36	0,17	0,28	0,36	0,01	0,96	0,0069	0,40
3	Milan Bergamo	0,0137	0,15	85849	0,17	0,31	0,24	0,42	0,54	0,01	0,98	0,0085	0,50
4	Stockholm-Bromma	0,0081	0,09	53259	0,10	0,24	0,04	0,02	0,02	0,01	0,77	0,0033	0,19
4	Växjö Kronoberg	0,0001	0,00	4072	0,01	0,20	0,01	0,00	0,00	0,01	0,57	0,0010	0,06
4	Ronneby	0,0000	0,00	5067	0,01	**1,00**	0,01	0,00	0,00	0,00	0,34	0,0007	0,04
4	Halmstad	0,0000	0,00	3205	0,01	**1,00**	0,01	0,00	0,00	0,00	0,34	0,0007	0,04
4	Ängelholm-Helsingborg	0,0000	0,00	7425	0,01	**1,00**	0,01	0,00	0,00	0,00	0,34	0,0007	0,04

En aquest cas no hi ha cap comunitat amb menys de 5 nodes i només trobem la comunitat 4 que agrupa nodes amb un alt coeficient d'agrupament però valors menyspreables per a la resta de centralitats. Tenim ara 3 comunitats amb nodes rellevants tant a la seva estructura interna com a la xarxa global.

4.4 Conjunts dominadors en potència de la xarxa

Per a trobar els conjunts dominadors en potència s'ha fet servir el codi de l'Annex 1. Aquest codi implementa un algorisme de *Threshold Accepting*, similar al descrit al final de capítol 2, per a trobar conjunts dominadors mínims o quasi-mínims.

En el cas de la xarxa completa, amb una dependència topològica molt important respecte dels nodes més grans, ha estat possible identificar 33 conjunts dominadors en potència mínims de 3 nodes. També hem comprovat totes les combinacions de parells de nodes descartant que hi hagi un conjunt mínim de 2 nodes.

Pel cas de la xarxa simplificada, que té una topologia més equilibrada, no hem estat capaços de trobar cap conjunt dominador en potència amb un nombre inferior a 21 nodes. Sense fer servir un algorisme exhaustiu no podem assegurar que no n'hi hagi cap amb menor nombre de nodes.

A la figura 4.7 visualitzem els nodes que componen totes les solucions trobades de 3 nodes de tota la xarxa, de color groc, i el conjunt de 21 nodes de la xarxa reduïda en color vermell.

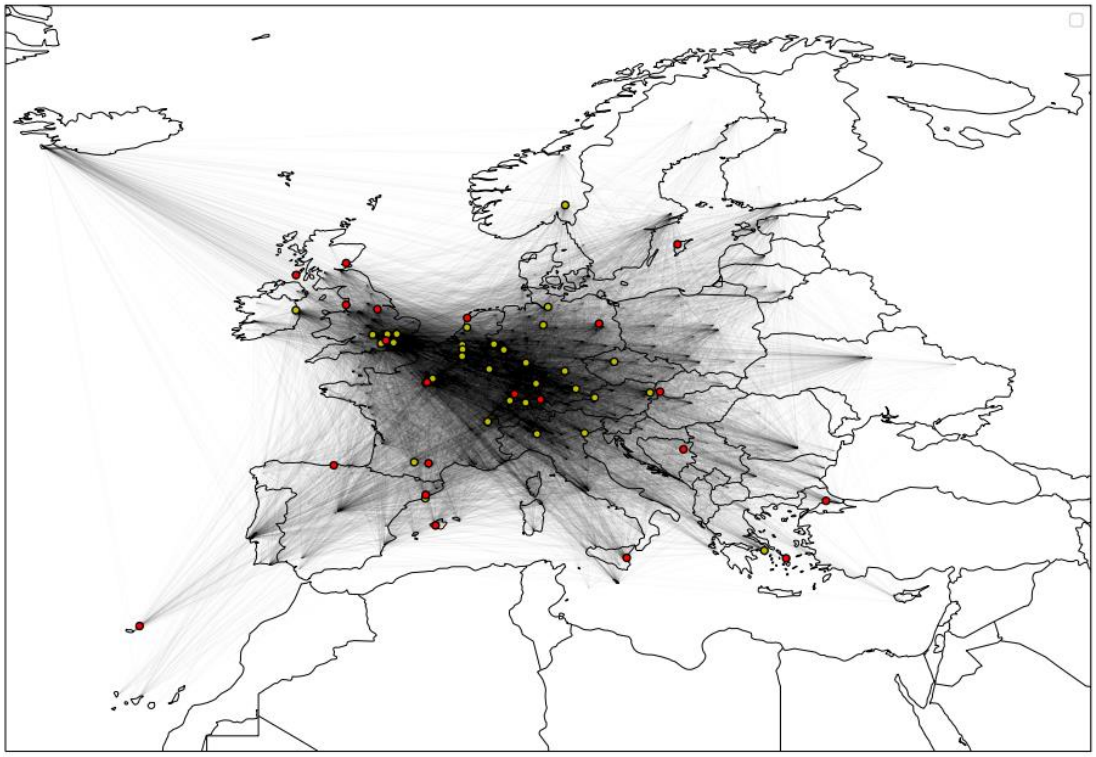

Fig. 4.7 Nodes dels conjunts dominadors en potència

Es pot apreciar com els nodes vermells de la xarxa simplificada es troben més repartits per la geografia europea, mentre que per a la xarxa completa totes les soluciones concentren els seus punts a la part central del continent.

Comparem les centralitats associades als conjunts dominadors d'ambdues xarxes (taules 4.4 i 4.5). A la primera taula apareixen els 35 nodes diferents que formen part d'alguna de les 33 solucions mínimes trobades, per ordre de major a menor centralitat d'intermediació. Destaquem amb color gris fosc els 2 aeroports que apareixen a tots els conjunts dominadors de 3 nodes. Aquests són Eleftherios Venizelos i Oslo Gardermoen, el primer líder en centralitat d'intermediació. Amb aquests dos i un altre qualsevol de la llista, amb coeficients prou importants però no tant destacats com els de la llista anterior, podem cobrir tota la xarxa segons el criteri de dominació en potència.

Si observem la segona taula, la 4.5, només existeix un conjunt amb centralitats rellevants, el corresponent a Londres Stansted. Els 20 aeroports restants no tenen cap indicador, excepte la centralitat de grau en alguns casos, que sigui destacable. Podem afirmar que la xarxa estaria dominada en potència per nodes que a priori no semblarien tan rellevants. Aquesta distribució podria indicar una dependència menor d'aquests grans aeroports d'interconnexió detectats a la xarxa completa.

Taula 4.4. Centralitats del conjunt dominador en potència de la xarxa completa

Aeroport	Centralitat d'intermediació		Nombre d'operacions		Coeficient d'agrupament	Centralitat de grau	Centralitat de comunicabilitat d'intermediació		Centralitat de proximitat de flux de corrent		Centralitat de rang de pàgina	
	Total	Total/màxim	Total	Total/màxim			Total	Total/màxim	Total	Total/màxim	Total	Total/màxim
Eleftherios Venizelos International	0,0292	**1,00**	189681	0,37	0,48	0,364	0,5071	0,63	0,0202	0,97	0,0053	0,79
Paris-Le Bourget	0,0237	0,81	53955	0,11	0,34	0,652	0,8006	**1,00**	0,0207	**1,00**	0,0067	**1,00**
Amsterdam Schiphol	0,0231	0,79	508299	**1,00**	0,35	0,634	0,7941	0,99	0,0207	1,00	0,0067	1,00
Oslo Gardermoen	0,0211	0,72	251193	0,49	0,47	0,439	0,6318	0,79	0,0204	0,98	0,0053	0,79
Charles de Gaulle International	0,0204	0,70	482678	0,95	0,36	0,622	0,7820	0,98	0,0207	1,00	0,0065	0,97
Munich	0,0195	0,67	401847	0,79	0,38	0,598	0,7771	0,97	0,0207	1,00	0,0064	0,95
London Stansted	0,0188	0,64	188156	0,37	0,35	0,612	0,7680	0,96	0,0207	1,00	0,0063	0,94
Brussels	0,0185	0,63	232719	0,46	0,37	0,602	0,7712	0,96	0,0207	1,00	0,0063	0,93
Zürich	0,0178	0,61	263549	0,52	0,35	0,644	0,8005	1,00	0,0207	1,00	0,0065	0,97
London Luton	0,0149	0,51	135083	0,27	0,38	0,600	0,7760	0,97	0,0207	1,00	0,0061	0,91
Cologne Bonn	0,0128	0,44	138832	0,27	0,40	0,557	0,7448	0,93	0,0206	0,99	0,0057	0,85
Frankfurt am Main	0,0126	0,43	475535	0,94	0,39	0,585	0,7717	0,96	0,0207	1,00	0,0060	0,89
Farnborough	0,0125	0,43	25787	0,05	0,42	0,524	0,7160	0,89	0,0206	0,99	0,0054	0,80
Vienna International	0,0119	0,41	240095	0,47	0,40	0,573	0,7665	0,96	0,0206	1,00	0,0058	0,86
Stuttgart	0,0117	0,40	117993	0,23	0,40	0,553	0,7389	0,92	0,0206	0,99	0,0056	0,84
Václav Havel Prague	0,0116	0,40	144013	0,28	0,39	0,593	0,7757	0,97	0,0207	1,00	0,0059	0,88
Dublin	0,0114	0,39	222326	0,44	0,44	0,500	0,6975	0,87	0,0205	0,99	0,0053	0,78
Geneva Cointrin International	0,0113	0,39	183591	0,36	0,39	0,587	0,7735	0,97	0,0207	1,00	0,0059	0,87
Hannover	0,0105	0,36	62439	0,12	0,42	0,506	0,6868	0,86	0,0205	0,99	0,0052	0,78
Toulouse-Blagnac	0,0103	0,35	98991	0,19	0,44	0,445	0,6050	0,76	0,0204	0,99	0,0047	0,70
Brussels South Charleroi	0,0102	0,35	51066	0,10	0,43	0,429	0,5664	0,71	0,0204	0,98	0,0045	0,68
Barcelona International	0,0101	0,34	323470	0,64	0,41	0,547	0,7429	0,93	0,0206	0,99	0,0055	0,82
London Biggin Hill	0,0095	0,32	15564	0,03	0,42	0,539	0,7374	0,92	0,0206	0,99	0,0054	0,80
Hamburg	0,0092	0,31	154478	0,30	0,44	0,516	0,7225	0,90	0,0206	0,99	0,0052	0,78
Luxembourg-Findel International	0,0086	0,29	70665	0,14	0,43	0,530	0,7299	0,91	0,0206	0,99	0,0053	0,79
EuroAirport Basel-Mulhouse-Freiburg	0,0069	0,23	78800	0,16	0,44	0,510	0,7143	0,89	0,0205	0,99	0,0050	0,75
Milano Linate	0,0059	0,20	116066	0,23	0,46	0,500	0,7123	0,89	0,0205	0,99	0,0049	0,73
Friedrichshafen	0,0056	0,19	14538	0,03	0,51	0,394	0,5648	0,71	0,0203	0,98	0,0040	0,59
Nuremberg	0,0053	0,18	53395	0,11	0,48	0,461	0,6625	0,83	0,0205	0,99	0,0046	0,68
Venice Marco Polo	0,0051	0,17	92144	0,18	0,47	0,478	0,6877	0,86	0,0205	0,99	0,0047	0,70
Oxford (Kidlington)	0,0046	0,16	807	0,02	0,48	0,386	0,5257	0,66	0,0203	0,98	0,0039	0,58
Toussus-le-Noble	0,0040	0,14	9910	0,02	0,44	0,305	0,3408	0,43	0,0200	0,96	0,0033	0,49
Salzburg	0,0038	0,13	29888	0,06	0,50	0,463	0,6787	0,85	0,0205	0,99	0,0045	0,67
Antwerp International (Deurne)	0,0037	0,13	1396	0,03	0,49	0,407	0,5805	0,73	0,0203	0,98	0,0041	0,60
Mönchengladbach	0,0034	0,12	6158	0,01	0,51	0,411	0,5968	0,75	0,0203	0,98	0,0041	0,61

Taula 4.5. Centralitats del conjunt dominador en potència de la xarxa simplificada

Aeroport	Centralitat d'intermediació		Nombre d'operacions		Coeficient d'agrupament	Centralitat de grau	Centralitat de comunicabilitat d'intermediació		Centralitat de proximitat de flux de corrent		Centralitat de rang de pàgina	
	Total	Total/màxim	Total	Total/màxim			Total	Total/màxim	Total	Total/màxim	Total	Total/màxim
London Stansted	0,0898	**1,00**	188156	0,37	0,22	0,436	0,7625	0,98	0,0107	**1,00**	0,0172	**1,00**
Gloucestershire	0,0062	0,07	2739	0,01	0,29	0,023	0,0067	0,01	0,0067	0,63	0,0023	0,13
Villacoublay-Vélizy (BA 107) Base aèria	0,0002	0,00	4861	0,01	0,73	0,017	0,0046	0,01	0,0066	0,61	0,0012	0,07
Maribor	0,0000	0,00	2035	0,00	0,67	0,009	0,0005	0,00	0,0047	0,44	0,0008	0,05
Federico Fellini International	0,0000	0,00	3576	0,01	0,33	0,009	0,0010	0,00	0,0048	0,45	0,0007	0,04
Toulouse/Francazal	0,0000	0,00	1297	0,00	0,00	0,003	0,0001	0,00	0,0023	0,21	0,0005	0,03
Skelleftea	0,0000	0,00	4746	0,01	0,00	0,003	0,0003	0,00	0,0023	0,21	0,0005	0,03
Umea	0,0000	0,00	18820	0,04	0,00	0,003	0,0000	0,00	0,0022	0,20	0,0006	0,04
Ronneby	0,0000	0,00	5067	0,01	1,00	0,006	0,0003	0,00	0,0037	0,34	0,0007	0,04
Limnos	0,0000	0,00	3422	0,01	0,00	0,003	0,0002	0,00	0,0023	0,21	0,0005	0,03
Kerry	0,0000	0,00	4137	0,01	0,00	0,003	0,0003	0,00	0,0023	0,21	0,0005	0,03
RAF Mildenhall	0,0000	0,00	4743	0,01	0,00	0,003	0,0000	0,00	0,0013	0,12	0,0010	0,06
ÃƒÂ–rebro	0,0000	0,00	6106	0,01	0,00	0,003	0,0001	0,00	0,0023	0,21	0,0005	0,03
Mariehamn	0,0000	0,00	3112	0,01	0,00	0,003	0,0003	0,00	0,0023	0,21	0,0005	0,03
Ängelholm-Helsingborg	0,0000	0,00	7425	0,01	1,00	0,006	0,0003	0,00	0,0037	0,34	0,0007	0,04
Kalmar	0,0000	0,00	6320	0,01	1,00	0,006	0,0003	0,00	0,0037	0,34	0,0007	0,04
Merville-Calonne	0,0000	0,00	1301	0,00	0,00	0,003	0,0001	0,00	0,0023	0,21	0,0005	0,03
Mikonos	0,0000	0,00	14076	0,03	0,00	0,003	0,0002	0,00	0,0023	0,21	0,0005	0,03
La Palma	0,0000	0,00	17070	0,03	1,00	0,006	0,0004	0,00	0,0034	0,31	0,0008	0,05
Madrid Getafe Base aèria	0,0000	0,00	1477	0,00	1,00	0,009	0,0002	0,00	0,0045	0,42	0,0009	0,05
De Kooy	0,0000	0,00	15908	0,03	0,00	0,003	0,0004	0,00	0,0023	0,21	0,0005	0,03

4.5 Fallades en cascada a la xarxa

Per a contrastar les principals diferències entre els dos models de graf considerats i acabar de detectar les principals vulnerabilitats de les xarxes estudiarem les caigudes en cascada.

Al capítol 3 s'ha definit el model per a simular l'increment de càrrega i la capacitat màxima dels nodes veïns quan un element de la xarxa cau segons les propietats de centralitat de cada node. Farem servir el model de Wang i Rong [10] per a simular fallades en cascada a les dues xarxes d'estudi.

Modelarem les caigudes segons tres variables bàsiques: el paràmetre $\propto$ **(alpha)** que regula la influència de les càrregues inicials, el **tipus de centralitat** utilitzada per al còmput inicial i el **tipus dels nodes iniciadors de la fallada en cascada** segons la seva càrrega inicial o si són de conjunts dominadors en potència. Es presenta una taula amb el detall dels paràmetres simulats.

Taula 4.6. Paràmetres estudiats en les fallades en cascada

Paràmetre	**Valors**	**Interpretació**
$\propto$	0.1	Valors més alts indiquen major influència de les càrregues inicials
	0.4 (Inclosos a annex III)	
	0.7 (Inclosos a annex IV)	
	0.9	
Tipus de centralitat	Operacions anuals (apartats 4.5.1 i 4.5.2)	Mesura de la centralitat amb la que modelarem la càrrega inicial i capacitat del node
	Proximitat de flux de corrent (apartats 4.5.3 i 4.5.4)	
	Centralitat de grau (apartats 4.5.5 i 4.5.6)	
Tipus de nodes	Càrrega alta	Càrregues dels nodes segons la centralitat escollida o nodes dels conjunts dominadors
	Càrrega mitjana	
	Càrrega baixa	
	Dominadors en potència	

Cal recordar que les centralitats s'han definit a l'apartat 2.1 i que considerem com a nodes de càrrega alta els 10 que tenen major centralitat segons l'opció escollida. Els de càrrega baixa són els 10 amb valors més petits i els mitjans són els 10 que queden a la meitat entre la càrrega màxima i mínima de tota la xarxa.

Respecte a iniciar la fallada amb els nodes de conjunts dominadors no podrem seguir la mateixa estratègia per a la xarxa completa i la reduïda. Això es deu a que en el primer cas tenim conjunts dominadors de 3 elements i en el segon considerem un únic conjunt compost per 21 nodes. Per a les simulacions de la xarxa completa seleccionarem una de les solucions de 3 nodes d'entre les 33

diferents obtingudes i 3 nodes aleatoris de cadascun dels elements del conjunt de nodes de càrrega petita, mitjana i gran.

Per a la xarxa reduïda limitar-se a 3 nodes seria escàs, doncs el conjunt dominador mínim en té 21. Decidim seleccionar 10 dels 21 nodes del conjunt de manera aleatòria i seleccionar-ne també 10 per a cada conjunt amb càrregues petites, mitjanes i grans.

En ambdós casos s'han calculat 20 realitzacions diferents i es mostra el resultat de fer la mitjana de tots els casos computats.

4.5.1 Fallades per nombre d'operacions ($\propto$=0.1)

Inicialment es mostra la tolerància a caigudes modelant la capacitat dels nodes de manera proporcional a les operacions aèries anuals. Aquesta capacitat és directament proporcional al paràmetre T esmentat al capítol 3 i a les gràfiques de la figura 4.8 es mostra el tant per 1 de nodes que cauen quan inhabilitem els 3 nodes escollits per a la xarxa completa o els 10 per a la reduïda.

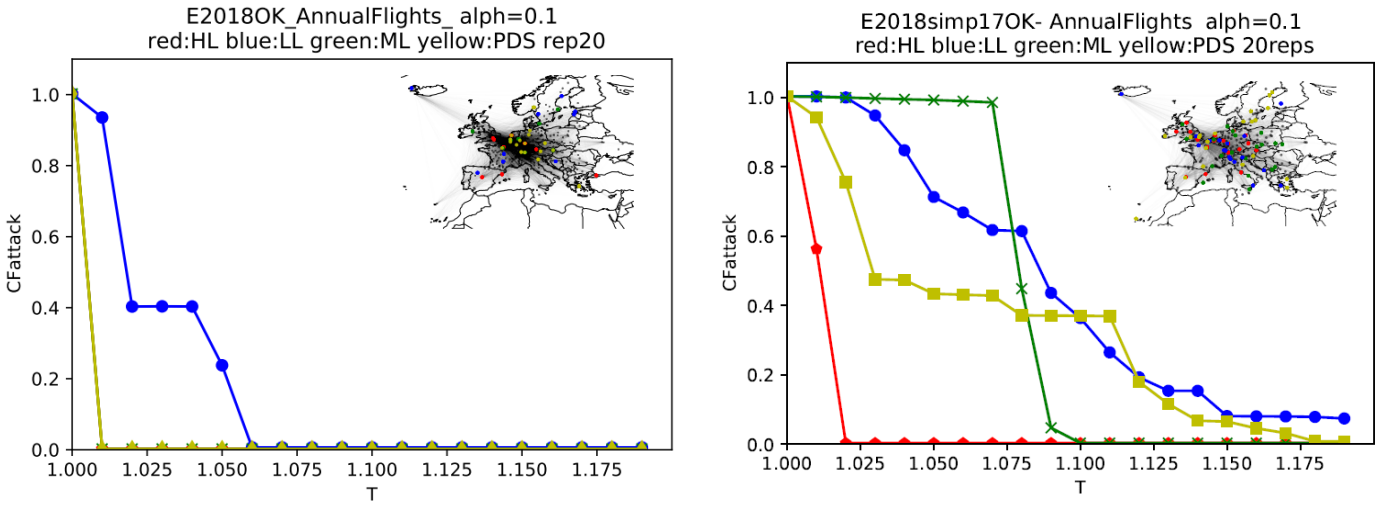

Fig. 4.8 Fallada en cascada amb capacitat proporcional al nombre d'operacions i $\propto$=0.1 a la xarxa completa (esquerra) i a la simplificada (dreta)

Així, la gràfica de l'esquerra de la figura 4.8 mostra 4 corbes diferenciades segons si els 3 nodes atacats són de càrrega alta, mitjana, petita o dominadors en potència amb colors vermell, blau, verd i groc, respectivament. Cal notar que aquestes corbes són per un valor d'$\propto$=0.1. Veurem més endavant la influència d'aquest paràmetre en els resultats.

Per a un valor de T=1 (cap node suporta càrregues addicionals a la pròpia) tots el nodes de la xarxa cauen (Cf=1). Quan comencem a incrementar el valor de T

per sobre d'1, observem que les corbes corresponents a fallades inicials en nodes de càrrega alta, mitjana i als conjunts dominadors en potència recuperen la integritat de la xarxa (Cf=0) ràpidament. Per a fer front a les caigudes inicials en nodes de càrrega petita, ens cal una tolerància a la fallada més alta (un valor més alt de T) respecte als casos anteriors. Els atacs als nodes de càrrega petita seran llavors els més nocius per a la xarxa completa amb aquesta configuració.

Quan s'analitzen els atacs de la mateixa manera a la xarxa simplificada (Figura 4.8 dreta) la xarxa continua sent més robusta quan els nodes atacats són els de càrrega gran mentre les altres 3 corbes mostren que cal més capacitat d'absorció de càrrega addicional per a recuperar el funcionament total. Ara són les corbes dels dominadors en potència i les de càrrega petita les que corresponen a caigudes més fortes, doncs necessiten una T major per a arribar a Cf=0.

També s'aprecia que l'increment de capacitat necessari per a ser robustos a caigudes és superior a la xarxa reduïda, el que es podria explicar per dues raons:

- els nodes atacats són 10 (a la xarxa completa només cauen 3)
- els nodes considerats tenen major influència sobre la xarxa en general per haver-ne simplificat l'estructura, eliminant prop de 200 nodes. Aquests nodes més superflus podrien ajudar a reabsorbir el tràfic.

4.5.2 Fallades per nombre d'operacions (∝=0.9)

El següent resultat a explorar seran les fallades per a un valor d'∝=0.9. S'han simulat també els valors d'∝ 0.4 i 0.7 per a cada apartat presentat a la memòria, però s'han inclòs als annexos III i IV per a facilitar la lectura.

Al treball de Wang i Rong sobre fallades en cascada [10] s'aprecia com per a valors petits d'∝ els nodes que produeixen més fallades són els de càrrega més baixa i per a valors alts són els de major càrrega. Comprovem el mateix comportament a la nostra xarxa.

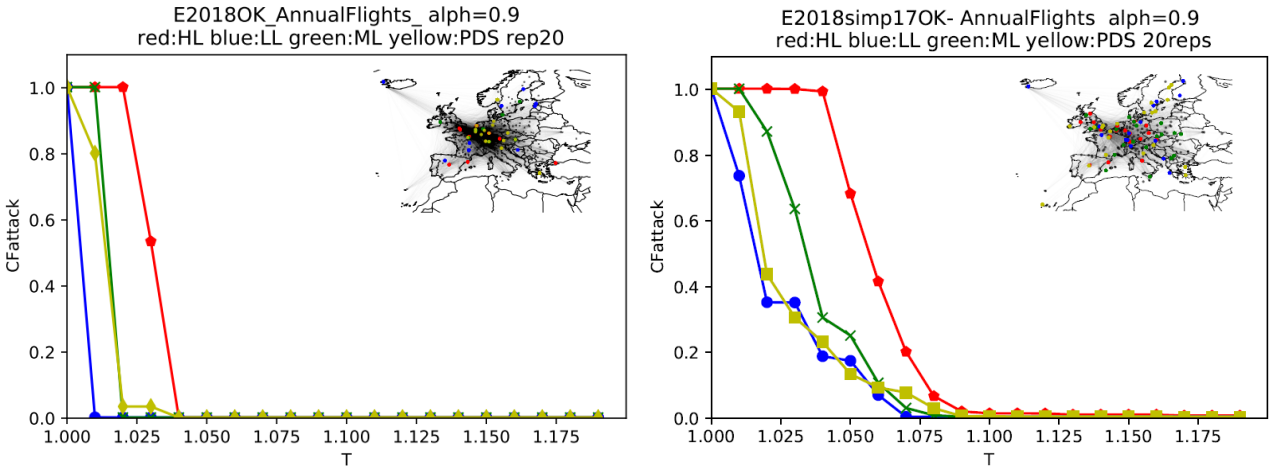

Fig. 4.9 Fallada en cascada amb capacitat proporcional al nombre d'operacions i ∝=0.9 a la xarxa completa (esquerra) i a la simplificada (dreta)

La xarxa completa és més susceptible als atacs sobre els nodes de càrrega alta i més robusta respecte a les caigudes d'elements de càrrega petita. Pels nodes de càrrega mitjana i conjunts dominadors els resultats es troben en un punt mitjà. Quan considerem la simplificació, tenim fallades més greus també amb els nodes de càrrega gran, però ara els menys susceptibles són els nodes de càrrega baixa i els del conjunt dominador només per a valors de T baixos.

4.5.3 Fallades per proximitat de flux de corrent (∝=0.1)

De manera anàloga analitzarem les fallades pels casos extrems d'∝, però ara utilitzant la mesura de proximitat de flux de corrent que no té en compte només el nombre d'operacions, sinó que considera informació de la topologia de la xarxa. Comencem pel cas d'∝=0.1 per a la xarxa completa.

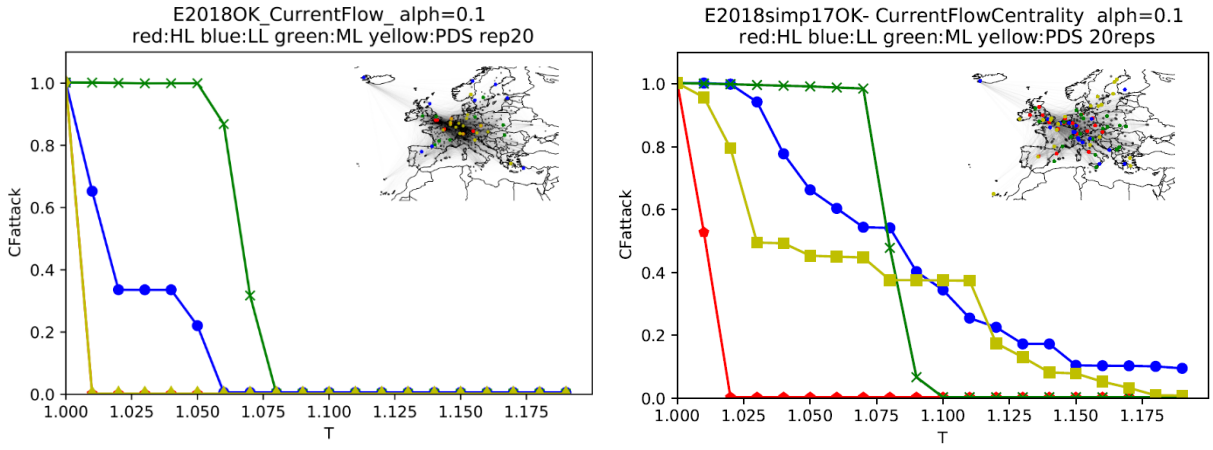

Fig. 4.10 Fallada en cascada amb capacitat proporcional a la proximitat de flux de corrent i ∝=0.1 a la xarxa completa (esquerra) i a la simplificada (dreta)

S'aprecia en aquest cas que la xarxa completa és més robusta a fallades amb atacs als nodes de càrrega alta i dominadors en potència i necessita una tolerància més alta per a conservar la integritat de la xarxa quan s'ataquen el nodes de càrrega petita i mitjana, especialment aquest darrers.

Quan considerem la xarxa simplificada els nodes més sensibles són els de càrrega petita i els conjunts dominadors i els més resistents els de càrregues mitjanes i especialment els de càrregues grans.

4.5.4 Fallades per proximitat de flux de corrent (∝=0.9)

Interessa comprovar ara si pel model de capacitat proporcional a la proximitat de flux de corrent el comportament també s'inverteix quan passem de valors baixos d'∝ a valors alts.

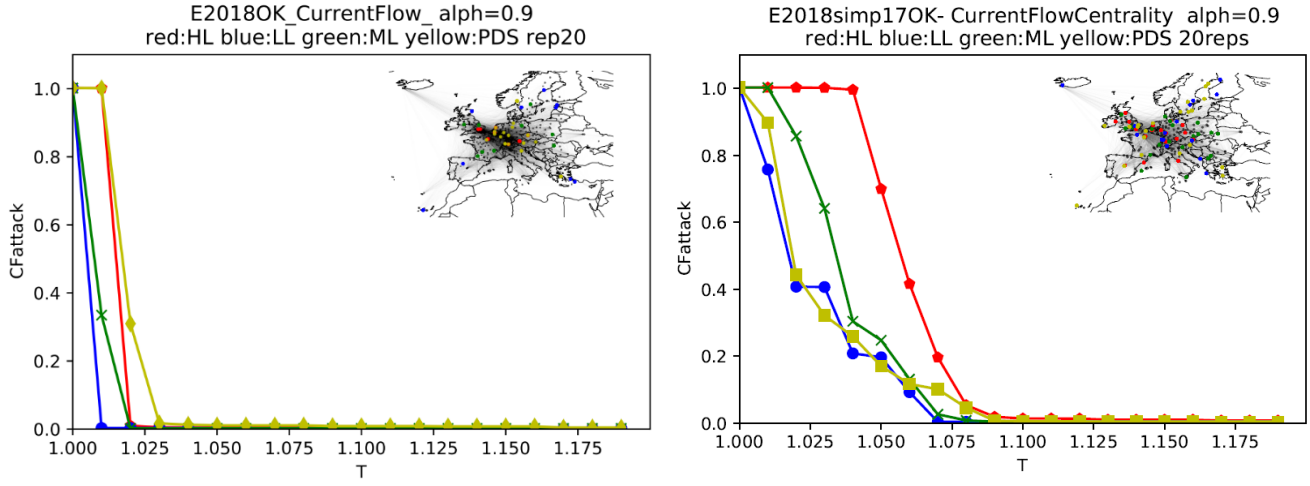

Fig. 4.11 Fallada en cascada amb capacitat proporcional a la proximitat de flux de corrent i ∝=0.9 a la xarxa completa (esquerra) i a la simplificada (dreta)

Examinem d'inici els resultats a la xarxa completa. Efectivament en aquest cas la xarxa cau amb més dificultat quan l'atac es realitza als nodes petits, independentment del tipus de xarxa estudiada.

Si analitzem els tipus que provoquen majors caigudes, trobem que són els dominadors en potència per a la xarxa completa i els de càrrega alta per a la simplificada. En ambdós casos els nodes de càrrega alta són molt limitants, tal com passava amb el nombre d'operacions i ∝=0.9

4.5.5 Fallades per centralitat de grau (∝=0.1)

Per cloure l'estudi es mostra com afecten els atacs quan la capacitat es modela amb la centralitat de grau, que únicament depèn de la topologia de la xarxa.

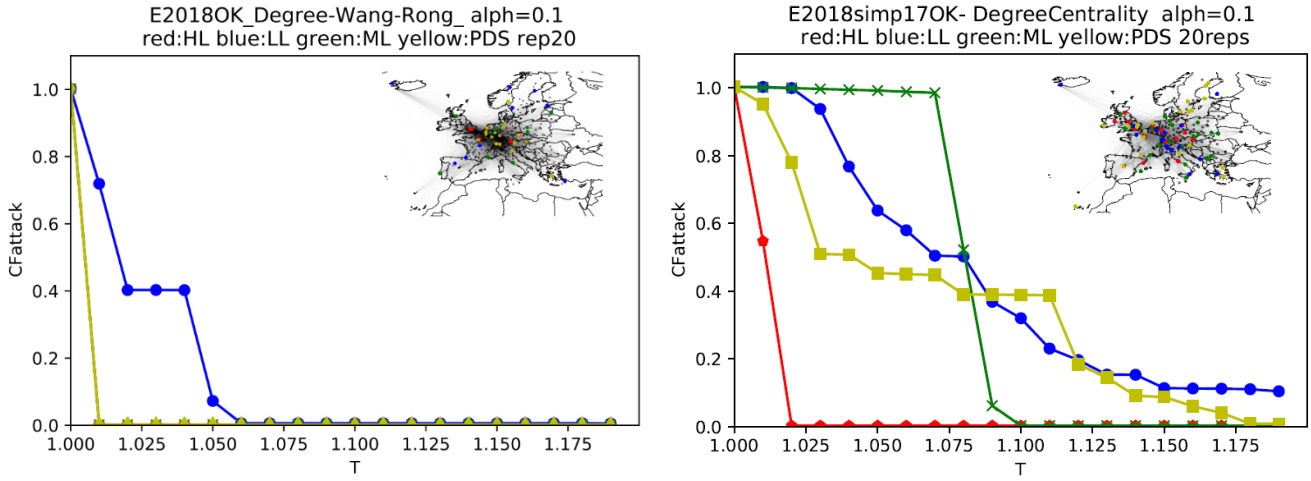

Fig. 4.12 Fallada en cascada amb capacitat proporcional a la centralitat de grau i $\propto$=0.1 a la xarxa completa (esquerra) i a la simplificada (dreta)

Les dues xarxes experimenten caigudes importants en rebre atacs els nodes de càrrega baixa. Per a la xarxa completa és indiferent si els atacs són a nodes de càrregues mitjanes, grans o de conjunts dominadors, mentre la xarxa simplificada és més resistent quan fallen els nodes de càrrega alta.

4.5.6 Fallades per centralitat de grau ($\propto$=0.9)

Novament quan augmentem $\propto$ a un valor alt de 0.9 s'inverteix la tendència i els atacs als nodes de càrrega alta passen a ser els més nocius per ambdues xarxes d'estudi.

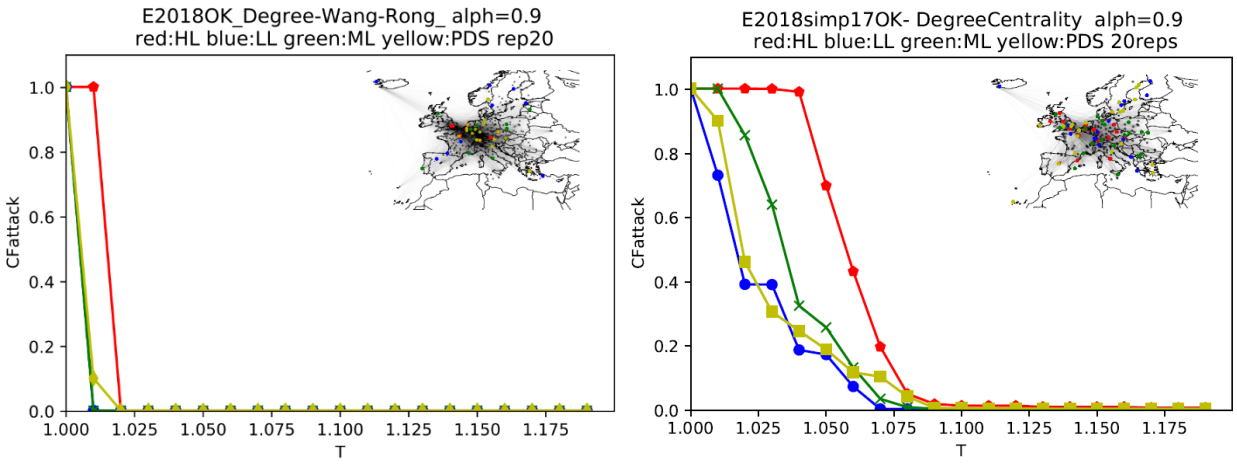

Fig. 4.13 Fallada en cascada amb capacitat proporcional a la centralitat de grau i $\propto$=0.9 per a la xarxa completa (esquerra) i a la simplificada (dreta)

La pèrdua de nodes de càrrega petita i mitjana és la que menys afecta a les dues xarxes considerades.

4.5.7 Resum de fallades en cascada segons paràmetre utilitzat

Resumim el comportament segons cada paràmetre per a cada xarxa a les taules 4.7 i 4.8. A aquestes indiquem per a cada elecció de paràmetre $\propto$ i tipus de centralitat la magnitud de fallada segons el tipus de node caigut. S'assignen nombres de l'1 al 4 a cada tipus, dels que provoquen majors fallades (1) al tipus amb menors fallades (4). Es considera una major fallada a la corba que necessita un valor més alt de T per a recuperar la integritat de la xarxa arribant a Cf=0. CA correspon a Càrrega Alta, CM a Càrrega Mitjana, CP a Càrrega Petita i DP a Dominador en Potència.

Taula 4.7. Fallades a la xarxa en funció dels paràmetres per a la xarxa completa

Centralitat	$\propto$	**CA**	**CM**	**CP**	**DP**
Nombre d'operacions	0.1	2	2	1	2
	0.9	1	3	4	2
Proximitat de flux de corrent	0.1	3	1	2	3
	0.9	2	3	4	1
Centralitat de grau	0.1	2	2	1	2
	0.9	1	3	3	2

Taula 4.8. Fallades a la xarxa en funció dels paràmetres per a la xarxa simplificada

Centralitat	$\propto$	**CA**	**CM**	**CP**	**DP**
Nombre d'operacions	0.1	4	3	1	2
	0.9	1	3	4	2
Proximitat de flux de corrent	0.1	4	3	1	2
	0.9	1	3	4	2
Centralitat de grau	0.1	4	3	1	2
	0.9	1	3	4	2

En general els nodes més crítics solen ser els de càrrega més alta per a valors d'$\propto$ alts o els de càrrega més baixa per a valors d'$\propto$ petits, independentment de la centralitat considerada. El nodes dels conjunts dominadors no són els més crítics pràcticament en cap cas, però tenen una influència molt gran en pràcticament tots els escenaris, essent gairebé sempre el segon tipus més limitant en produir fallades en cascada.

CONCLUSIONS

L'anàlisi d'una xarxa com la dels aeroports europeus no és una feina trivial, com tampoc ho és destriar la informació rellevant del soroll en una xarxa d'aquesta complexitat. Per aquest motiu hem optat per estudiar en paral·lel la xarxa amb tota la informació disponible sobre nombre d'operacions anuals a cada aeroport i una altra de simplificada que elimina els enllaços amb poca rellevància. Tal i com hem vist, aquesta reducció ha ajudat a determinar grups d'aeroports (o comunitats) consistents -similars en ambdós casos- i els nodes més rellevants del conjunt. També s'ha comprovat que la xarxa completa és més resistent a fallades en tots els casos.

En relació a la cerca de conjunts dominadors en potència, veiem que a la xarxa completa ha estat possible trobar 33 solucions mínimes de 3 nodes que aconsegueixen dominar tota la xarxa de més de 500 nodes. En canvi, per a la solució reduïda no hem establert cap conjunt dominador de menys de 21 nodes, el que indicaria que aquesta darrera xarxa té menys dependència dels grans aeroports d'interconnexió centre-europeus.

En quant a les fallades en cascada s'ha reproduït el mateix comportament de l'article base referenciat [10], on es mostra que la xarxa elèctrica dels EEUU i Canadà és més vulnerable a la caiguda de nodes amb càrrega menor quan el paràmetre $\propto$ (que controla la influència de la càrrega inicial dels nodes) és petit. En contraposició, en considerar valors d'$\propto$ grans, els nodes de càrregues baixes són els que menys fallades en cascada provoquen. Considerant la importància dels nodes dels conjunts dominadors, s'ha modelat la caiguda d'aquests per veure si la seva influència és superior a la de la resta de nodes i les fallades en cascada són més pronunciades. El resultat final ha estat que la influència és prou important, però similar a la dels nodes de càrrega alta per a la xarxa completa.

Podem concloure llavors que els conjunts dominadors en potència podrien tenir la seva rellevància a l'estudi de les xarxes d'aeroports, tot i no haver-se trobat que en el nostre estudi afegeixin gaire més informació que la que ja teníem amb la classificació per càrregues. Cal fer notar que el tipus de centralitat utilitzada per a modelar les fallades ha fet variar els resultats, però no ha estat determinant en cap dels casos per a canviar radicalment els comportaments observats sense la identificació d'aquests nodes.

BIBLIOGRAFIA

[1] Barabási, A., “Network science”, Cap. 2 en *Graph Theory*, Autoeditat. pp 1-14 (2014).

[2] Wang, J., Mo, H. et al., “Exploring the network structure and nodal centrality of China’s air transport network: A complex network approach”, Journal of Transport Geography 19, 712-721 (2011).

[3] Newman, M., “Measures and metrics”, Cap. 7 en *Networks*, Oxford University Press, Oxford pp 158-217 (2018).

[4] Estrada, E., Higham, D.J. and Hatano, N. “Communicability betweenness in complex networks”, Physica A, 764-774 (2009).

[5] Newman, M., “A measure of betweenness centrality based on random walks”, Social Networks 27, 39-54 (2005).

[6] Estrada, E., Knight, P., “Spectral Node Centrality”, Cap. 15 en *A first course in network theory*, Oxford University Press, Oxford pp 157-169 (2015).

[7] Clauset, M., Newman, M. et al., “Finding community structure in very large networks”, Physical Review E 70, 068111 (2004).

[8] Haynes, T.W., Hedetniemi, S.M. et al. “Domination in graphs applied to electric power networks”, Siam J. Discrete Math, Vol.15, No. 4, 519-529 (2002).

[9] Dueck, G. and Scheuer, T., “Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing”, Journal of Computational Physics 90, 161-175 (1990).

[10] Wang, J. and Rong, L., “Cascade-based attack vulnerability on the US power grid”, Safety Science 47, 1332-1336 (2009).

[11] Wu, Z.X., Peng, G. Et al, “Cascading failure spreading on weighted heterogeneous networks”, J. Stat. Mech., 5-13 (2008).

[12] Li, P., Wang, B.H. et al., “A limited resource model of fault-tolerant capability against cascading failure of complex network”, Eur. Phys. J. B 62, 1 (2008).

[13] Gelabert, P., “Analysis of the European airport network as a complex network”, UPCommons, (2018).

ANNEXOS

Annex I. Codi per a la cerca dels conjunts dominadors

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Filename: MPDS_det.py
# Sergio Fernández Bertolín -
# Fitxer per trobar un Power Dominating Set


def MPDS(G, T_0, T_min, MAX_ITER, coef):
    """Retorna el conjunt de nodes a
    *graf* que conformen un Power Dominating Set mínim o quasi-mínim

    Resulta de cercar la millor solució amb un Threshold Acceptance aplicat
    sobre els nodes que encara no s'han assolit amb les soluciones trobades

    La solució és incremental i afegeix un nou node cada cop que completa el
    màxim d'iteracions sense trobar un Power Dominating Set

    Paràmetres
    ----------
    G : Graf de NetworkX
        Graf no dirigit.

    T_0 : Tolerància màxima permesa al procés de Threshold Acceptance
    T_min : Tolerància final permesa al procés de Threshold Acceptance
    MAX_ITER : Nombre màxim d'iteracions del Threshold Acceptance

    coef: Coeficient de refredament del Threshold Acceptance

    Retorna
    -------
    solucio
        Conjunt de nodes que conformen la millor solució trobada aplicant
        Threshold Acceptance.
    pds
        1 si el conjunt de nodes retornat és un Power Dominating Set.
        Percentatge de nodes assolits pel conjunt `solucio` en cas contrari.

    """

    import time
    import grinpy as gp
    import random
```

```
def modificacio(nodes, no_assolits):
    """Selecciona aleatòriament un node del conjunt de nodes `no_assolits`
    i l'afegeix al conjunt `nodes` per a cerca un Power Dominating Set.

    Paràmetres
    ----------
    nodes : list, set
        Darrera millor solució acceptada cercant un Power Dominating Set.
    no_assolits : list, set
        El conjunt de nodes que encara no hem assolit amb la darrera
        millor solució acceptada a la cerca del Power Dominating Set.

    Retorna
    -------
    nodes : list, set
        El nou conjunt de nodes a testejar com a Power Dominating Set

    """
    nou_node=random.sample(no_assolits,1)
    nodes.append(nou_node[0])
    return nodes


def es_power_dominating_set_(G, nodes):
    """Retorna el percentatge de nodes a
    *G* que són accessibles des del conjunt `nodes`
    Paràmetres
    ----------
    G : Graf de NetworkX
        Graf no dirigit.
    nodes : list, set
        Contenidor iterable de nodes a G.

    Retorna
    -------
    float
        1 si els nodes al conjunt `nodes` tenen conjunt 1-forcing a *G*.
        Percentatge de nodes assolits en cas contrari.

    """
    nodes=gp.set_closed_neighborhood(G,nodes)
    Z = set(n for n in nodes if n in G)
    while gp.is_k_forcing_active_set(G, Z, 1):
        Z_temp = Z.copy()
        for v in Z:
            if gp.is_k_forcing_vertex(G, v, Z, 1):
                Z_temp |= set(gp.neighborhood(G, v))
        Z = Z_temp
    return len(Z)/len(G.nodes())
```

```
def nodes_no_assolits(G, nodes):
    """Retorna el conjunt de nodes a
    *G* que no són accessibles des del conjunt `nodes`

    Paràmetres
    ----------
    G : Graf de NetworkX
        Graf no dirigit.
    nodes : list, set
        Contenidor iterable de nodes a G.

    Retorna
    -------
    list
        Conjunt de nodes no assolits pel conjunt 'nodes'.

    """
    nodes=gp.set_closed_neighborhood(G,nodes)
    Z = set(n for n in nodes if n in G)
    Y = list(G.nodes)
    while gp.is_k_forcing_active_set(G, Z, 1):
        Z_temp = Z.copy()
        for v in Z:
            if gp.is_k_forcing_vertex(G, v, Z, 1):
                Z_temp |= set(gp.neighborhood(G, v))
        Z = Z_temp

    for node in Z:
        Y.remove(node)
    return Y


def TA_inc(G, T_0, T_min, MAX_ITER, coef):
    """Retorna el conjunt de nodes a
    *graf* que conformen un Power Dominating Set mínim o quasi-mínim

    Resulta de cercar la millor solució amb un Threshold Acceptance aplicat
    sobre els nodes que encara no s'han assolit amb les soluciones trobades

    La solució és incremental i afegeix un nou node cada cop que completa el
    màxim d'iteracions sense trobar un Power Dominating Set

    Paràmetres
    ----------
    G : Graf de NetworkX
        Graf no dirigit.

    T_0 : Tolerància màxima permesa al procés de Threshold Acceptance
```

```
T_min : Tolerància final permesa al procés de Threshold Acceptance
MAX_ITER : Nombre màxim d'iteracions del Threshold Acceptance

coef: Coeficient de refredament del Threshold Acceptance

Retorna
-------
solucio
   Conjunt de nodes que conformen la millor solució trobada aplicant
   Threshold Acceptance.
pds
   1 si el conjunt de nodes retornat és un Power Dominating Set.
   Percentatge de nodes assolits pel conjunt `solucio` en cas contrari.

"""

inici=time.time()
itera=True

initial_nodes=random.sample(list(G),1)
solucio_nova=initial_nodes.copy()
nova_solucio_millor=solucio_nova.copy()
no_assolits=nodes_no_assolits(G,nova_solucio_millor)

while itera:
   C_0=es_power_dominating_set_(G, nova_solucio_millor)
   T=T_0
   if C_0==1:
      itera=False
      break
   while T>T_min:
      for iter in range(MAX_ITER):
         C_nova=es_power_dominating_set_(G, solucio_nova)
         if C_nova > C_0 or C_0-C_nova < T:
            C_0=C_nova
            nova_solucio_millor=solucio_nova.copy()
            if C_0==1:
               T=T_0
               itera=False
               break
         solucio_nova.pop()
         solucio_nova=modificacio(solucio_nova, no_assolits)
      T=T*coef
   print(nova_solucio_millor)
   print(len(nova_solucio_millor)," nodes")
   print(es_power_dominating_set_(G, nova_solucio_millor))
   no_assolits=nodes_no_assolits(G,nova_solucio_millor)
   solucio_nova=modificacio(nova_solucio_millor, no_assolits)

final=time.time()
```

```
        print(final-inici," segons")
        print(nova_solucio_millor)
        print(len(nova_solucio_millor)," nodes")
        print(es_power_dominating_set_(G, nova_solucio_millor))
        return nova_solucio_millor, es_power_dominating_set_(G,
nova_solucio_millor)


    inici=time.time()
    solucio, pds= TA_inc(G, T_0, T_min, MAX_ITER, coef)

    final=time.time()
    print(final-inici," segons")

    return solucio, pds
```

Annex II. Codi per a modelar fallades en cascada

```
# -*- coding: utf-8 -*-
#!/usr/bin/python
# (c) Francesc Comellas 2021
# Modificat per Sergio Fernández Bertolín
#


from __future__ import print_function
import time
import itertools
import statistics
import random as rnd
from random import seed
import networkx as nx
from pylab import *
import pickle

#import grinpy as gp

from  math import log
import matplotlib.pyplot as plt
import numpy as np
from collections import Counter

from matplotlib.cbook import flatten

import cartopy.crs as ccrs
import cartopy.feature as cf


print('works with Python 3.7.6 i NetworkX 2.4\n')
import sys
print('PYTHON version',sys.version)
print('NX version',nx.__version__)

## http://stackoverflow.com/questions/10919664/averaging-list-of-lists-python

starttime = time.time()

pds_comp=[[233, 304, 133, 10],[304, 10, 136, 267],[304, 206, 74, 10],
        [304, 250, 10, 206],[304, 87, 10, 261],[122, 304, 38, 10],
        [258, 26, 304, 10],[435, 304, 10, 255],[304, 10, 158, 132],
        [258, 10, 15, 304],[304, 62, 318, 10],[304, 10, 80, 72],
        [304, 10, 301, 206],[304, 10, 174, 258],[304, 122, 10, 452],
        [304, 10, 350, 322],[304, 10, 280, 468],[304, 412, 305, 10],
        [304, 399, 10, 264],[304, 262, 410, 10],[304, 264, 303, 10],
        [304, 54, 10, 13],[304, 158, 10, 79],[304, 10, 81, 163],
        [304, 13, 10, 308],[304, 475, 10, 0],[304, 338, 467, 10],
```

```
        [304, 176, 165, 10],[304, 240, 10, 134],[304, 10, 453, 494],
        [304, 339, 10, 233],[304, 305, 10, 289],[304, 297, 438, 10],
        [304, 392, 10, 468],[304, 489, 10, 282],[304, 280, 10, 254],
        [304, 10, 433, 350],[304, 350, 10, 436]]

pds_simp=[236, 235, 186, 331, 19, 148, 107, 60, 342, 35, 42, 277, 262, 234, 325,
336, 260, 34, 195, 272, 3]

#####################################
### funcio inicialitza
### calcula  la carrega inicial Lj
### de cada node. Depen d'alpha !!
#####################################

def ini_carr(alpha,tol):
    for j in g:
        kj=gload[j]
        v=g.neighbors(j)
        sumkm=0
        for m in v:
            sumkm=sumkm+gload[m]
        Lj=(kj*sumkm)**alpha
        g.nodes[j]['Lj']=round(Lj,4)
    carr_max(tol)

#####################################
### funcio carr_maxima
### per calcular la carrega m‡xima,
### Cj de cada node. Depen de T !!
#####################################

def carr_max(T):
    for j in g:
        Cj=(T*g.nodes[j]['Lj'])
        g.nodes[j]['Cj']=round(Cj,4)



#####################################
### funcio distrib_carr
### redistribueix la carrega del node
### al veins que encara no han fallat
#####################################

def distrib_carr(i):
    sum_carr_veins=0
    for j in g.neighbors(i):
        if g.nodes[j]['OK']==1:
            sum_carr_veins=sum_carr_veins+g.nodes[j]['Lj'] #si tots els veïns OK=0?
    for j in g.neighbors(i):
```

```
        if g.nodes[j]['OK']==1 and sum_carr_veins!=0:
            newLj=g.nodes[j]['Lj']+g.nodes[j]['Lj']*g.nodes[i]['Lj']/sum_carr_veins # si
Lj=0; newLj=Lj(antiguo)
            g.nodes[j]['Lj']=round(newLj,4)

#####################################
### produeix llista nodes
###  de min a max Cj
#####################################

def listdownup():
    gsort=sorted(g.nodes(data=True), key=lambda labels: labels[1]['Cj'])
##          gsort=sorted(g.nodes_iter(data=True),          key=lambda          labels:
labels[1]['Cj'],reverse=True)
    ordlist=[]
    for i in range(len(gsort)):
        nod=gsort[i][0]
        ordlist.append(nod)
    return ordlist

#### Per invertir la llista downup
##   updown=downup.reverse()

## Per fer una llista amb els 10 primer nodes d una llista
## [cjlist[i] for i in range(10)]
##



#####################################
### funcio cascada
### a partir d'un node fa la cascada
### i retorna la llista dels que han
### fallat
### la llista es una llista de llistes
### a cada posicio diu els nodes que
### han fallat al pas 0,1,2,etc
#####################################

def cascada(v):
    eccg=nx.eccentricity(g)
    ecc=eccg[v]
    for i in g:
        g.nodes[i]['OK']=1
    g.nodes[v]['OK']=0
    dist=-1
    finit=0
    oldlevel=[v]
    hanfallat=[oldlevel]
    while (finit==0):
```

```
        dist=dist+1
        newlevel=[]
        for pos in oldlevel:
            distrib_carr(pos) #new neighbour's node load
            for j in g.neighbors(pos): #which nodes are failing?
                if g.nodes[j]['OK']==1:
                    if ((g.nodes[j]['Lj']) >= (g.nodes[j]['Cj'])):
                        g.nodes[j]['OK']=0
                        newlevel.append(j)
        hanfallat.append(newlevel)
        oldlevel=newlevel
        if (dist==ecc):
            finit=1
    return hanfallat


################

def FCcommunicability_exp(G):
    import scipy.linalg
    nodlst = list(G) # ordering of nodes in matrix
    A = nx.to_numpy_matrix(G,nodlst)
    # convert to 0-1 matrix
    # convert to 0-1 matrix
    A[A!=0.0] = 1
    # communicability matrix
    expA = scipy.linalg.expm(A.A)
    mapping = dict(zip(nodlst,range(len(nodlst))))
    c = {}
    for u in G:
        c[u]={}
        for v in G:
            c[u][v] = float(expA[mapping[u],mapping[v]])
    return c


# ########################################
# Communicability distance index (big Gamma)

def comm_dist_idx(G):
    Gamma_idx=0.0
    cdst=FCcommunicability_exp(G)
    for u in G:
        for v in G:
            val=cdst[u][u]+cdst[v][v]-2*cdst[u][v]
            val=abs(val)
            Gamma_idx+=sqrt(val)
    return Gamma_idx/2

# ########################################
```

```
# Wiener index

def wiener_idx(G, weight=None):
    # compute sum of distances between all node pairs
    # (with optional weights)
    wiener=0.0
    if weight is None:
        for n in G:
            path_lengths=list(nx.single_source_shortest_path_length(G,n).values())
            wiener+=sum(path_lengths)
    else:
        for n in G:

path_lengths=list(nx.single_source_dijkstra_path_length(G,n,weight=weight).val
ues())
            wiener+=sum(path_lengths)
    return wiener/2.0

# ########################################
# Q index

def Q_idx(G,wieneridx):
    return log(0.857763885*wieneridx/comm_dist_idx(G))



def standardDev(alist):
    theMean = mean(alist)

    sum = 0
    for item in alist:
        difference = item - theMean
        diffsq = difference ** 2
        sum = sum + diffsq

    sdev = math.sqrt(sum/(len(alist)-1))
    return sdev

#####################################
###  Part principal
###
#####################################

CFlist20LL=[]
CFlist20HL=[]
CFlist20ML=[]
CFlistPDS=[]
```

```
centrality= ' AnnualFlights '
#centrality= ' DegreeCentrality '
#centrality= ' CurrentFlowCentrality '

seed(1)

reps=20


print( "\\\ ATENTION: centrality:"+centrality+"   repeticions: ",reps,"\n")

alfa=0.1
alpha='0_1'
dt = 0.01
ngraus=10  # considera sols 10 vertexs  de cada tipus per fer la caiguda
tolvals = arange(1.00, 1.20, dt)
granslist=[]
petitslist=[]
miglist=[]
pdslist=[]
gransnodelist=[]
petitsnodelist=[]
mignodelist=[]
pdsnodelist=[]

corr=[]
diameter=[]
avgdist=[]
clust=[]
wiener=[]
qidx=[]
histgrm=[]

hstgrmlenmax=0

for idx in range(reps):
    print('\n\n ',idx,'\n')

    # Seleccio de xarxa simplificada
    grafname='E2018simp17OK'

    # Selecció de xarxa completa
    #grafname='E2018OK'
    g=nx.read_gml(grafname+'.gml',label='id')



    g.name=grafname+str(idx)

    g.name=grafname
```

```
    # Càlcul de la centralitat corresponent per a cada tipus de càrrega
    gload=nx.current_flow_closeness_centrality(g)

    # Càrrega, centralitat de grau
    #gload=nx.degree_centrality(g)

    # Càrrega, nombre d'operacions
    # gload={}
    # for node in g:
    #     gload[node]=g.nodes[node]['annualFlights']



    print(time.asctime( time.localtime(time.time()) ))
    print( 'file:',grafname+str(idx)+'.edges')

    print( 'alpha:',alfa)
    print(  nx.info(g))
    print ('============================')


    degs=g.degree


################
#   GRANS
################

    CFlist=[]
    tol=1.00

    ini_carr(alfa,tol) #Assigna una carrega a cada node (tot actius)
    while (tol<1.20):

        if len(CFlist)==0:
            load=listdownup()    #sort nodes, saves nodes names
            load.reverse()
            load=load[0:20]
            ##    print graus

            grans=rnd.sample(load,ngraus)
            print ('\nGRANS '+str(grans))
            gransload=[]
            for i in grans:
                gransload.append(round(gload[i],4))
            print(gransload)
#       print( '\nGRANS '+str(grans))
#       print( ' alfa:',alfa,' T: ',tol)
```

```
        start = time.time()
        faillist=[]
        #f.write('\n GRANS \nTolerance: '+str(tol))
        for i in grans:
            ini_carr(alfa,tol)
            fail=cascada(i) #nodes that failed
            flat=list(itertools.chain.from_iterable(fail)) #give all nodes in 'fail'
            #print flat
            #f.write('\n'+str(flat))
            CFi= len(flat)
            faillist.append(CFi)
            elapsed = (time.time() - start)
#         print( 'Nodes failing: '+str(faillist) )
        CFattck=round(sum(faillist)/(ngraus*(g.order()-1)*1.0),4)
#         print( 'CFattck: ',CFattck)
        CFlist.append(CFattck)
        #print time.asctime( time.localtime(time.time()) )
        #print elapsed
        tol=tol+dt
    CFlist20HL.append(CFlist)
    gransnodelist.append(grans)

    Lj0=[]
    Ljn=0
    for i in grans:
        Ljn=g.nodes[i]['Lj']
        Lj0.append(Ljn)
    inf=zip(grans, gransload,Lj0)

    granslist.append(inf)


################
#   PETITS
################

    CFlist=[]
    tol=1.00
    ini_carr(alfa,tol)

    while (tol<1.20):
        if len(CFlist)==0:
            load=listdownup()
            #graus.reverse()
            ##    print graus
            load=load[0:20]
            ##    print graus

            petits=rnd.sample(load,ngraus)
            print ('\nGPETITS '+str(petits))
```

```
            petitsload=[]
            for i in petits:
                petitsload.append(round(gload[i],4))
            print(petitsload)

#         print( '\nPETITS '+str(petits))
#         print( ' alfa:',alfa,' T: ',tol)


        start = time.time()
        faillist=[]
        #f.write('\n PETITS \nTolerance: '+str(tol))
        for i in petits:
            ini_carr(alfa,tol)
            fail=cascada(i)
            flat=list(itertools.chain.from_iterable(fail))
            #print flat
            #f.write('\n'+str(flat))
            CFi= len(flat)
                #if CFi==1 and ftol[petits.index(i)]==0: #get node tolerance
                #    ftol[petits.index(i)]=round(tol,4)
            faillist.append(CFi)
            elapsed = (time.time() - start)
#         print( 'Nodes failing: '+str(faillist))
        CFattck=round(sum(faillist)/(ngraus*(g.order()-1)*1.0),4)
#         print( 'CFattck: ', CFattck)
        CFlist.append(CFattck)
        #print time.asctime( time.localtime(time.time()) )
        #print elapsed
        tol=tol+dt
    CFlist20LL.append(CFlist)
    petitsnodelist.append(petits)

    Lj0=[]
    Ljn=0
    for i in petits:
        Ljn=g.nodes[i]['Lj']
        Lj0.append(Ljn)

    inf=zip(petits, petitsload, Lj0)

    petitslist.append(inf)

################
# MITJANS
################


    CFlist=[]
    tol=1.00
```

```
ini_carr(alfa,tol)
while (tol<1.20):

    if len(CFlist)==0:

        load=listdownup()
        load.reverse()    #from highest to lowest
        grausload=[]
        for i in load:
            grausload.append(gload[i])

        ##    print graus
        #hstgrm=nx.degree_histogram(g)
        #load_max=gload[load[0]]
        load_mig=gload[load[round(len(load)/2)]]


        for i in range(len(load)):
            ns=grausload[i]-load_mig
            if ns<0:
                if abs(grausload[i]-load_mig)-abs(grausload[i-1]-load_mig)>0:
                    i=i-1 #We choose the closest value to load_mig
                break

        #range of mitjans
        mitjans=[]
        j=i
        if i!=len(load)-1: #if i=99; middle node is the lowest node
            mitjans.append(load[i])
        else:
            mitjans=petits

        while len(mitjans)<20:
                ns=abs(grausload[i-1]-load_mig)-abs(grausload[j+1]-load_mig)
                if ns<0 and i>0: #we choose the closest values to load_mig
                    mitjans.append(load[i-1])
                    i=i-1
                    #if i==0:
                    #    mitjans=grans
                elif ns>=0 and j<(len(load)-1):
                    mitjans.append(load[j+1])
                    j=j+1
                    if j==99:
                        mitjans=petits

        # print graus
        # load=load[0:20]
        # ##    print graus

        # petits=rnd.sample(load,4)
```

```
            # print ('\nGPETITS '+str(petits))
            # petitsload=[]
            # for i in petits:
            #     petitsload.append(round(gload[i],4))
            # print(petitsload)
            mitjans=rnd.sample(mitjans,ngraus)


            mitjansload=[]
            for i in mitjans:
                mitjansload.append(round(gload[i],4))


#         print( ' alfa:',alfa,' T: ',tol)


        start = time.time()

        faillist=[]
        #f.write('\n MITJANS \nTolerance: '+str(tol))
        for i in mitjans:
            ini_carr(alfa,tol)
            fail=cascada(i)
            flat=list(itertools.chain.from_iterable(fail))
            #print flat
            #f.write('\n'+str(flat))
            CFi= len(flat)
            faillist.append(CFi)
            elapsed = (time.time() - start)
#         print( 'Nodes failing: '+str(faillist))
        CFattck=round(sum(faillist)/(ngraus*(g.order()-1)*1.0),4)
#         print( 'CFattck: ',CFattck)
        CFlist.append(CFattck)
        tol=tol+dt
    CFlist20ML.append(CFlist)
    mignodelist.append(mitjans)

    print( '\nMITJANS '+str(mitjans))
    print(mitjansload)

    Lj0=[]
    Ljn=0
    for i in mitjans:
        Ljn=g.nodes[i]['Lj']
        Lj0.append(Ljn)

    inf=zip(mitjans, mitjansload, Lj0)

    miglist.append(inf)
    #f.close()
```

```
###############

#   Preparem llista per a nodes del PDS (elecció aleatòria)

#################

    CFlist=[]
    tol=1.00
    ini_carr(alfa,tol)

    #pds_comp=[10, 304, 0, 13, 15, 35, 47, 50, 58, 62, 72, 87, 99, 122, 147, 153,
156, 163, 206, 214, 233, 240, 258, 264, 267, 305, 335, 350, 357, 361, 374, 377,
435, 438, 468]
    #pds_simp=[236, 235, 186, 331, 19, 148, 107, 60, 342, 35, 42, 277, 262, 234,
325, 336, 260, 34, 195, 272, 3]

    PDS=rnd.sample(pds_simp,ngraus)
    print(PDS)


    while (tol<1.20):
        if len(CFlist)==0:
            #load=gload.copy()
            #graus.reverse()
            ##    print graus
            #pdsl=rnd.sample(PDS,4)
            pdsl=PDS

            print('pdsl ',pdsl)
            pdsload=[]
            for i in pdsl:
                pdsload.append(round(gload[i],4))

#       print( '\nPETITS '+str(petits))
#       print( ' alfa:',alfa,' T: ',tol)


        start = time.time()
        faillist=[]
        #f.write('\n PETITS \nTolerance: '+str(tol))
        for i in pdsl:
            ini_carr(alfa,tol)
            fail=cascada(i)
            flat=list(itertools.chain.from_iterable(fail))
            #print flat
            #f.write('\n'+str(flat))
            CFi= len(flat)
                #if CFi==1 and ftol[petits.index(i)]==0: #get node tolerance
                #    ftol[petits.index(i)]=round(tol,4)
```

```
            faillist.append(CFi)
            elapsed = (time.time() - start)
#         print( 'Nodes failing: '+str(faillist))
        CFattck=round(sum(faillist)/(ngraus*(g.order()-1)*1.0),4)
#         print( 'CFattck: ', CFattck)
        CFlist.append(CFattck)
        #print time.asctime( time.localtime(time.time()) )
        #print elapsed
        tol=tol+dt
    CFlistPDS.append(CFlist)
    pdsnodelist.append(pdsl)

    Lj0=[]
    Ljn=0
    for i in pdsl:
        Ljn=g.nodes[i]['Lj']
        Lj0.append(Ljn)

    inf=zip(pdsl, pdsload, Lj0)

    pdslist.append(inf)

###############



# AVGdiameter=statistics.mean(diameter)
# AVGavgdist=round(statistics.mean(avgdist),4)
# AVGclust=round(statistics.mean(clust),4)
# AVGwiener=statistics.mean(wiener)
# AVGqidx=round(statistics.mean(qidx),4)
#  AVGcorr=round(nx.degree_assortativity_coefficient(g),4)  #  random  regular
AVGcorr=1
# AVGhistgrm=[]

# for i in range(hstgrmlenmax):
#     hist=[]
#     for j in range(len(histgrm)):
#         if len(histgrm[j])>i:
#             hist.append(histgrm[j][i])
#         else:
#             hist.append(0)

#     AVGhistgrm.append(statistics.mean(hist))



###############
print
```

```
t = arange(1.00, 1.20, dt)

processtime = time.time()-starttime
print('\n running time = %-8.0f \n' % (processtime))

processtime = time.time()-starttime

f=open(grafname+centrality+'-alpha'+alpha+'--failure_info.txt','w')
f.write('\ngraf type: '+grafname)
f.write('\ncentrality: '+centrality+'\n')
f.write('\nreps: '+str(reps)+'\nalpha:'+str(alfa)+'\ngraus:'+str(ngraus)+'\n')
f.write('\nprocess time [s]: '+str(processtime)+'\n')
# f.write('\n gransini: '+str(granslist)+'\n petitssini: '+str(petitslist)+'\n')
f.write('\ntolvals: '+str(tolvals)+'\n')
# f.write('\n diameter: '+str(AVGdiameter)+'\n avg distance:'+str(AVGavgdist)+'\n
clustering:'+str(AVGclust)+'\n Wiener index:'+str(AVGwiener)+'\n Q
index:'+str(AVGqidx)+'\n Correlation:'+str(AVGcorr)+'\n')
# f.write('\n avg histogram: '+str(AVGhistgrm)+'\n')



f.write('\n\nHL\n[node '+centrality+' initial_load   L0j\n')
for ind in range(len(granslist)):
    f.write(str(set(granslist[ind])))
    f.write('\n')
f.write('\nFail vs tol \n'+str(CFlist20HL)+'\n')


f.write('\n\nLL\n[node '+centrality+' initial_load   L0j\n')
f.write('node - load - Loj\n')
for ind in range(len(petitslist)):
    f.write(str(set(petitslist[ind])))
f.write('\nFail vs tol \n'+str(CFlist20LL)+'\n')


f.write('\n\nML\n[node '+centrality+' initial_load   L0j\n')
for ind in range(len(miglist)):
    f.write(str(set(miglist[ind])))
f.write('\nFail vs tol \n'+str(CFlist20ML)+'\n')


f.write('\n\nPDS\n[node '+centrality+' initial_load   L0j\n')
for ind in range(len(pdslist)):
    f.write(str(set(pdslist[ind])))
f.write('\nFail vs tol \n'+str(CFlistPDS)+'\n')

f.close()


print( tolvals)
```

```
hl =  [round(float(sum(col))/len(col),4) for col in zip(*CFlist20HL)]
ll =  [round(float(sum(col))/len(col),4) for col in zip(*CFlist20LL)]
ml =  [round(float(sum(col))/len(col),4) for col in zip(*CFlist20ML)]
pd =  [round(float(sum(col))/len(col),4) for col in zip(*CFlistPDS)]
print( '------------------')
print( hl)
print( '------------------')
print( ll)
print( '------------------')
print( ml)
print( '------------------')
print( pd)
print( '------------------')
print
plot(tolvals, hl,'r-',marker='p')
plot(tolvals, ll,'b-',marker='o')
plot(tolvals, ml,'g-',marker='x')
plot(tolvals, pd,'y-',marker='s')
axis([1,1.2,0,1.1])
tcks = arange(1.00, 1.20, 0.025)
xticks(tcks)
xlabel('T')
ylabel('CFattack')
title(grafname+'-'+centrality+'  alph='+str(alfa)+'\n  red:HL  blue:LL  green:ML
yellow:PDS '+str(reps) +'reps')


# draw graph in inset
#plt.axes([0.55,0.15,0.30,0.30])

#####
# figure()
ax = plt.axes([0.55,0.55,0.30,0.30],projection=ccrs.PlateCarree())

ax.add_feature(cf.COASTLINE, lw=0.15)
ax.add_feature(cf.BORDERS, lw=0.1)

#####

import matplotlib.pyplot as plt
pos={}
for i in g:
    pos[i]=(g.nodes[i]['lon'],g.nodes[i]['lat'])
plt.axis('off')
nx.draw_networkx_nodes(g,pos,node_size=0.2,node_color='k',node_shape='o',
alpha=0.6)
nx.draw_networkx_edges(g,pos,alpha=0.01,width=0.001)

for i in faillist:
    pos[i]=(g.nodes[i]['lon'],g.nodes[i]['lat'])
```

```
gransall=list(flatten(gransnodelist))
mitjansall=list(flatten(mignodelist))
petitsall=list(flatten(petitsnodelist))
pdsall=list(flatten(pdsnodelist))

nx.draw_networkx_nodes(g,pos,node_size=1,nodelist=gransall,node_color='r',alpha=0.9)
nx.draw_networkx_nodes(g,pos,node_size=1,nodelist=mitjansall,node_color='g',alpha=0.9)
nx.draw_networkx_nodes(g,pos,node_size=1,nodelist=petitsall,node_color='b',alpha=0.9)
nx.draw_networkx_nodes(g,pos,node_size=1,nodelist=pdsall,node_color='y',alpha=0.9)



##
savefig(grafname+centrality+' alph='+str(alfa)+'rep'+str(reps)+'.pdf',dpi=1200)
show()


#####################################
```

Annex III. Fallades en cascada per a ∝=0.4

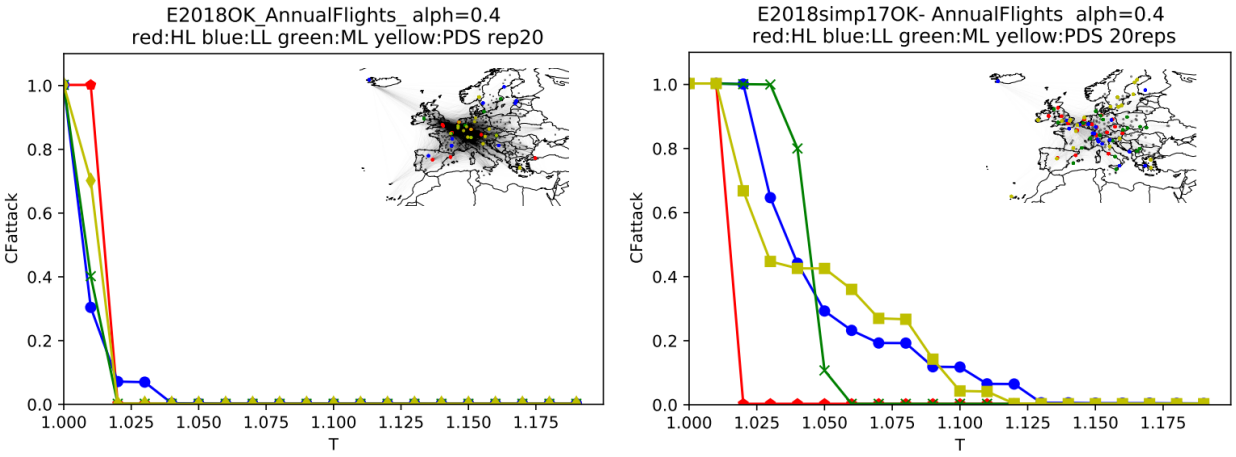

Fig. A.1 Fallada en cascada amb capacitat proporcional al nombre d'operacions i ∝=0.4 a la xarxa completa (esquerra) i a la simplificada (dreta)

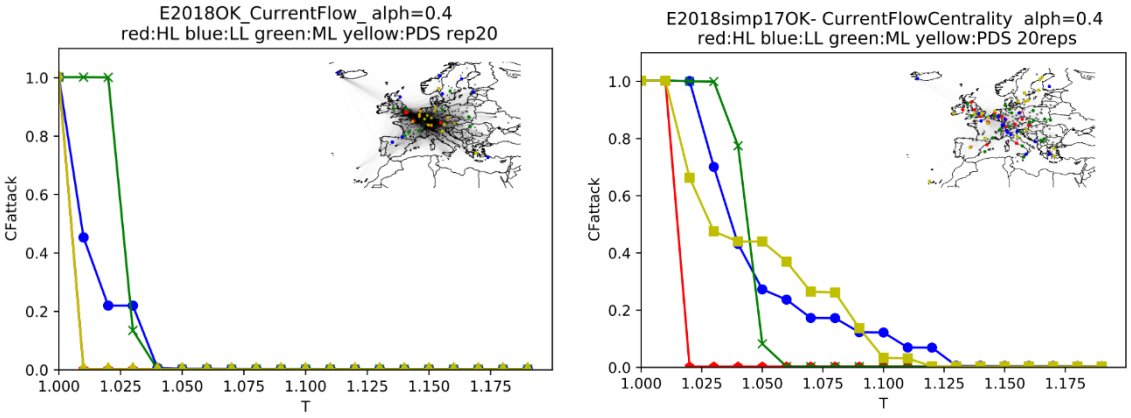

Fig. A.2 Fallada en cascada amb capacitat proporcional a la proximitat de flux de corrent i ∝=0.4 a la xarxa completa (esquerra) i a la simplificada (dreta)

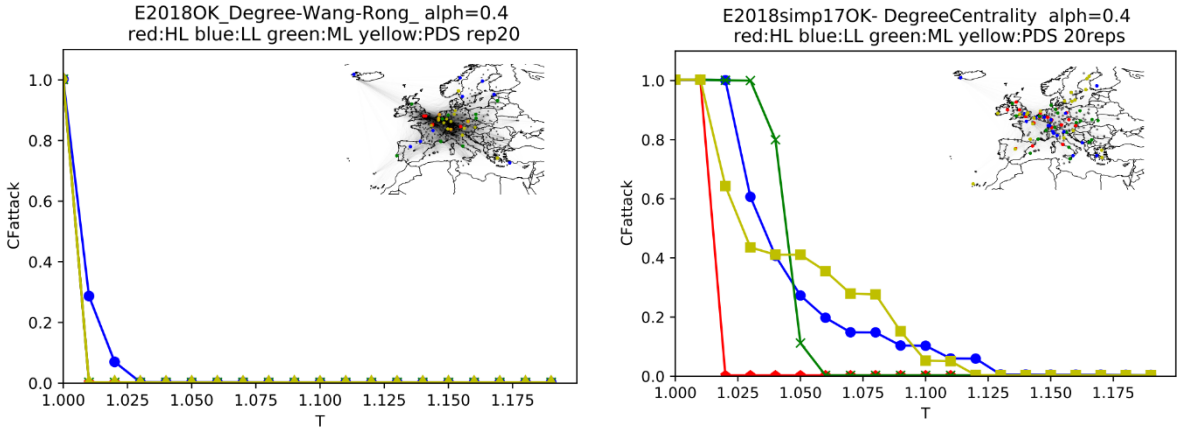

Fig. A.3 Fallada en cascada amb capacitat proporcional a la centralitat de grau i $\propto$=0.4 a la xarxa completa (esquerra) i a la simplificada (dreta)

Annex IV. Fallades en cascada per a ∝=0.7

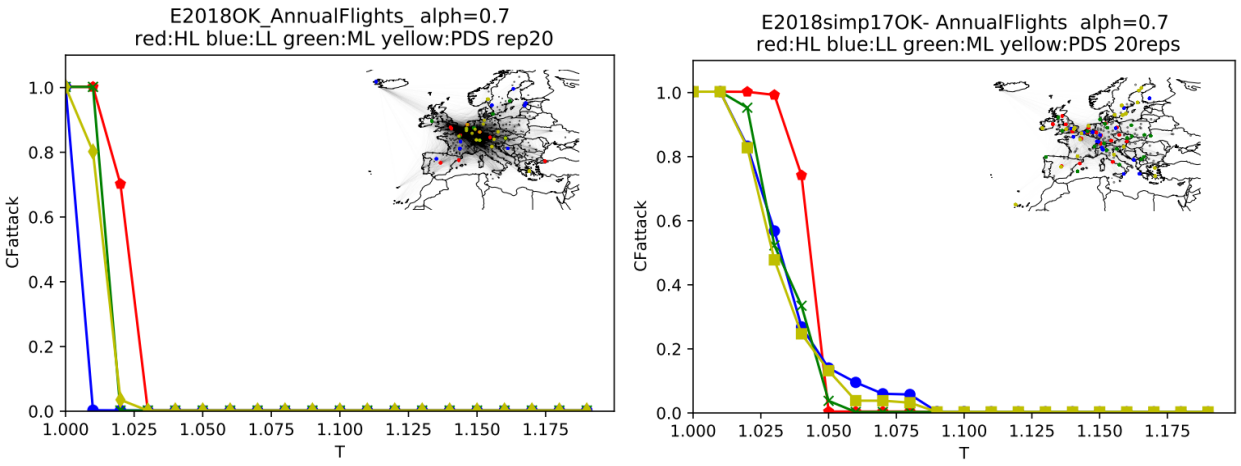

Fig. A.4 Fallada en cascada amb capacitat proporcional al nombre d'operacions i ∝=0.7 a la xarxa completa (esquerra) i a la simplificada (dreta)

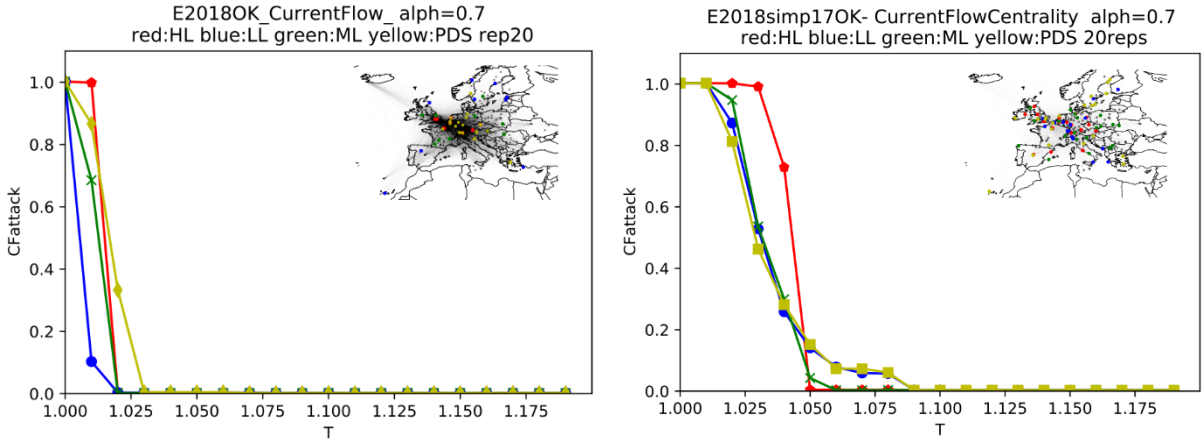

Fig. A.5 Fallada en cascada amb capacitat proporcional a la proximitat de flux de corrent i ∝=0.7 a la xarxa completa (esquerra) i a la simplificada (dreta)

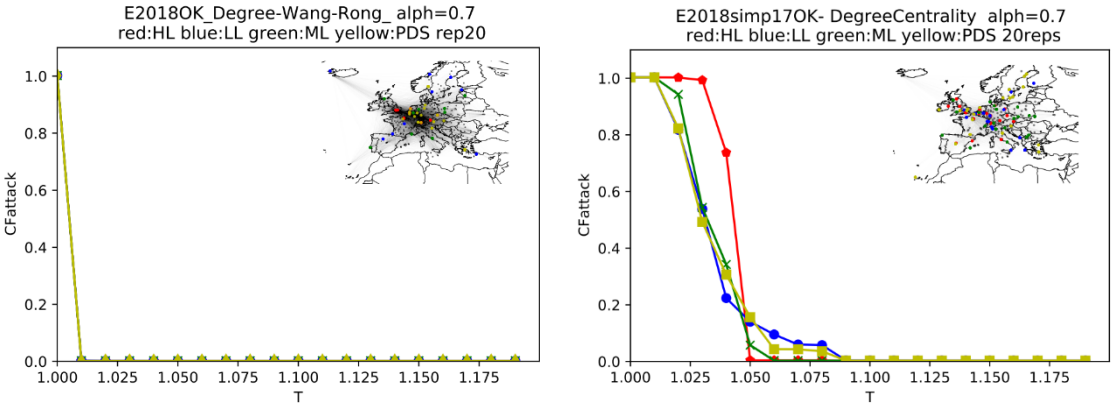

Fig. A.6 Fallada en cascada amb capacitat proporcional a la centralitat de grau i $\propto$=0.7 a la xarxa completa (esquerra) i a la simplificada (dreta)