Algorismia

Estudi Experimental de Connectivitat i Percolació de Grafs

Pau Belda, Guillem Cabré, Marc Peñalver, Prisca Oleart

Curs 2024-25, Quatrimestre de tardor

Continguts

1 Introducció

L'estudi de la connectivitat en grafs sota processos de percolació és un camp d'interès clau en diverses disciplines, incloent la teoria de grafs, la física estadística i l'anàlisi de xarxes complexes. La percolació modela situacions en què nodes o arestes d'un graf poden fallar de manera aleatòria, i aquestes fallades poden tenir un impacte significatiu en la connectivitat global del sistema. Un aspecte crucial en aquests estudis és la transició de fase, un fenomen en què una propietat estructural del graf, com ara la connexió entre els seus components, pateix un canvi sobtat a mesura que es varia un paràmetre, com la probabilitat de fallida.

En aquest estudi, ens centrarem en l'efecte de la percolació sobre diferents tipus de grafs, tant deterministes com aleatoris. Concretament, ens proposem investigar com la percolació de nodes i d'arestes afecta la connectivitat del graf i, més específicament, com es produeix la transició de fase que determina el pas d'un graf connex a un graf fragmentat en múltiples components. Les nostres hipòtesis inicials estan basades en la variació d'aquests fenòmens en funció del tipus de graf i del procés de percolació.

Aquestes hipòtesis ens permetran explorar com diferents configuracions estructurals i processos de fallida afecten la transició de fase i la connectivitat dels grafs.

2 Definicions

2.1 Percolació

La percolació en un graf G consisteix en eliminar o desactivar nodes o arestes, i posteriorment es mesura com això afecta una certa propietat global del graf. Quan desactivem una aresta o un node, direm que ha tingut una fallida.

En termes generals, l'objectiu és estudiar com el graf passa d'estar completament connectat a parcialment o totalment desconnectat a mesura que es treuen alguns dels seus components. Quan parlem de percolació, considerem una probabilitat p que determina si una component del graf (node o aresta) es desactiva aleatòriament. Aquest mecanisme és especialment rellevant per a l'estudi de xarxes complexes, ja que ens ajuda a comprendre com de robust o vulnerable és el sistema que volem analitzar.

- **Percolació per nodes**: Cada node té una probabilitat p de ser desactivat. Un cop fet això, s'analitza com ha canviat la connectivitat del graf.
- **Percolació per arestes**: En aquest cas, les arestes es desactiven en lloc dels nodes. Això també afecta la connectivitat, ja que les connexions directes entre nodes es perden.

Després d'aplicar el procés de percolació (sobre nodes o arestes), s'obté un graf percolat, que és la versió modificada del graf original, amb una connectivitat reduïda i, possiblement, components desconnectats. Aquest graf l'anomenarem G_{p}.

2.2 Transició de Fase

Una transició de fase d'un graf per a una propietat concreta Π fa referència a un resultat satisfactori d'un procés de percolació aplicat al graf. En el nostre cas aquesta propietat Π serà la connectivitat del graf.

Definim un resultat com a satisfactori si, donat que es troba una probabilitat de valor q tal que es compleix la propietat Π al graf G_q (definim aquesta probabilitat com q_Π), per als grafs $G_{q'}$ on $q' > q_\Pi$, aquests verifiquen la propietat Π, i als grafs $G_{q'}$ on $q' < q_\Pi$, no la verifiquen (ambdues afirmacions són vàlides si es compleixen amb una probabilitat prou alta).

Quan s'ha obtingut aquest resultat, diem que la propietat Π presenta una transició de fase al voltant de q_Π.

2.3 Objectius de la Experimentació

En aquest projecte, realitzem un estudi experimental sobre la transició de fase en grafs sotmesos a un procés de percolació, modelat mitjançant un paràmetre $p \in [0, 1]$ que representa la probabilitat que un node o aresta falli. L'objectiu principal és analitzar com varia el nombre de components connexes d'un graf durant el procés de percolació, avaluant l'existència d'un valor crític, conegut com a *threshold* o umbral de transició de fase, al voltant del qual el graf experimenta canvis significatius en la seva connectivitat.

Per a aquest anàlisi, considerem diversos models de grafs, incloent xarxes quadrades, grafs geomètrics aleatoris i altres models paramètrics. Els experiments es realitzaran per grafs de diferents mides, enfocant-nos en el comportament asimptòtic a mesura que el nombre de nodes creix. L'estudi es complementarà amb la implementació d'algorismes per a generar aquests grafs, aplicar percolació i calcular el nombre de components connexes.

Objectius específics:

- Estudiar la possible transició de fase en graelles quadrades $n = m \times m$, sent n el nombre de nodes, sota un procés de percolació per nodes i arestes amb probabilitat p. La hipòtesi plantejada és que la percolació de nodes generarà una transició de fase més ràpida que la percolació d'arestes.

La raó d'aquesta hipòtesi és que, en eliminar nodes, també s'eliminen totes les arestes associades a aquests nodes, fet que altera la connectivitat de manera més dràstica. Per tant, s'espera que la connectivitat del graf es vegi més afectada quan fallin nodes en comparació amb quan només fallin les arestes.

- Estudiar la possible transició de fase en grafs geomètrics aleatoris connexos (*Random geometric graphs*), sota un procés de percolació d'arestes.

  La nostra hipòtesi és que, a mesura que augmenta el radi de connexió, el graf esdevé més dens, fet que dificulta la fragmentació del graf quan es produeixen fallades. Per tant, preveiem que, a mesura que augmenta la densitat del graf, serà necessària una probabilitat de fallida més alta per provocar una transició de fase.

- Estudiar la possible transició de fase sota un procés de percolació d'arestes en graelles triangulars $n = \frac{rows \times (rows+1)}{2}$, sent n el nombre de nodes.

  La nostra hipòtesi és que, com més arestes connectin un node, més resistent serà el graf davant la percolació d'arestes, ja que la seva connectivitat global es veurà menys alterada. Això ens permetrà analitzar com el nombre d'arestes per node influeix en la capacitat del graf per mantenir la seva integritat davant fallades. Per validar aquesta hipòtesi, compararem la percolació per arestes en dos models: el graf de graella quadrada, on cada node està connectat a quatre veïns, i el graf de graella triangular, on cada node està connectat a sis veïns. Aquest contrast ens permetrà estudiar com la diferència en el nombre de connexions per node afecta la robustesa del graf.

- Estudiar la possible transició de fase sota un procés de percolació de nodes en un graf de tipus *Hub Graph* o graf Barabási-Albert.

  El graf de Barabási-Albert és un model que genera grafs amb una estructura de xarxa d'escala lliure, on alguns nodes (hubs) concentren un nombre desproporcionat d'arestes. La hipòtesi que proposem és que la tolerància d'aquesta xarxa a les fallades es veurà afectada principalment per la percolació de nodes. En concret, si els hubs comencen a fallar, la xarxa perdrà la seva connectivitat molt més ràpidament que en un graf més uniforme, ja que els hubs són crítics per mantenir el graf connectat.

Els resultats obtinguts permetran caracteritzar el comportament de diferents models de grafs sota condicions de percolació i explorar la robustesa d'aquestes xarxes davant de fallades aleatòries.

3 Grafs Seleccionats

En aquesta secció s'exposen els grafs seleccionats per a l'estudi experimental. Concretament, explicarem les peculiaritats de cada graf. Així mateix, es detallaran els algorismes utilitzats per a la generació de cada tipus de graf.

3.1 Graella Quadrada

Algorisme 1 Generació de Graf de Graella Quadrada $G(m \times m)$

Entrada: m (dimensió de la graella)
Sortida: Graf g

Inicialitzar $n = m \times m$ (nombre de nodes del graf)
Crear un graf buit g amb n nodes
for $i = 0$ fins a $m - 1$ **do**
    for $j = 0$ fins a $m - 1$ **do**
        $nodeActual = i \times m + j$
        if $i < m - 1$ **then**
            $nodeFilaInferior = (i + 1) \times m + j$
            Afegir una aresta entre $nodeActual$ i $nodeFilaInferior$
        end if
        if $j < m - 1$ **then**
            $nodeColumnaDreta = i \times m + (j + 1)$
            Afegir una aresta entre $nodeActual$ i $nodeColumnaDreta$
        end if
    end for
end for
Retornar el graf g

El generador de grafs de graella quadrada crea un graf amb una estructura regular en què cada node és adjacent als nodes de la fila superior, inferior, esquerra i dreta, si aquests existeixen. Això s'aconsegueix comprovant per cada node si existeixen nodes a la fila inferior i a la columna dreta, i si és així, s'afegeixen les arestes corresponents.

3.2 Graella Triangular

Algorisme 2 Generació de Graf de Graella Triangular $G(rows)$

Entrada: $rows$ (nombre de files)
Sortida: Graf g

```
Inicialitzar n = rows×(rows+1)/2 (nombre de nodes del graf)
Crear un graf buit g amb n nodes
nodeActual = 0
for i = 0 fins a rows − 1 do
    for j = 0 fins a i do
        if j < i then
            nodeDreta = nodeActual + 1
            Afegir una aresta entre nodeActual i nodeDreta
        end if
        if i < rows − 1 then
            nodeInferiorEsquerra = nodeActual + i + 1
            Afegir una aresta entre nodeActual i nodeInferiorEsquerra
            nodeInferiorDret = nodeActual + i + 2
            Afegir una aresta entre nodeActual i nodeInferiorDret
        end if
        nodeActual = nodeActual + 1
    end for
end for
Retornar el graf g
```

El generador de grafs de graella triangular crea un graf amb n nodes, on $n = 1 + 2 + 3 + \ldots + rows = \frac{rows \times (rows+1)}{2}$ per tal de conseguir una estructura triangular. Aquests nodes tenen una estructura en què cada node està connectat als veïns de la dreta i l'esquerra, així com als nodes superiors i inferiors, tant a l'esquerra com a la dreta, sempre que aquests existeixin. Això s'aconsegueix comprovant per a cada node si hi ha nodes a la fila inferior i a la columna dreta, i si és així, s'afegeixen les arestes corresponents.

3.3 Graf Geomètric Aleatori

Algorisme 3 Generació de Graf Geomètric Aleatori $G(n, r)$

Entrada: n (nombre de nodes), r (radi de connexió)
Sortida: Graf g

```
Crear un graf buit g amb n nodes
Inicialitzar un vector de coordenades coords de longitud n
for i = 0 fins a n − 1 do
    coords[i].x = rand01()
    coords[i].y = rand01()
end for
for i = 0 fins a n − 1 do
    for j = i + 1 fins a n − 1 do
        if distanciaEuclidiana(coords[i], coords[j]) < r then
            Afegir una aresta entre i i j al graf g
        end if
    end for
end for
Retornar el graf g
```

El generador de grafs geomètrics aleatoris crea un graf amb n nodes, on cada node es col·loca aleatòriament en un espai bidimensional unitari. Dos nodes estan connectats per una aresta si la distància entre ells és menor que un radi r especificat. Això es determina calculant la distància euclidiana entre tots els parells de nodes i afegint arestes quan la distància és inferior a r.

3.4 Graf de Barabási-Albert

Algorisme 4 Generació de Graf de Barabási-Albert $G(n, m_0, m)$

Entrada: n (nombre de nodes), m_0 (nombre de nodes inicials), m (grau de connexió per nou node)
Sortida: Graf g

Crear un graf buit g amb n nodes
Inicialitzar un vector *connection_degree* de longitud n
for $i = 0$ fins a $m_0 - 1$ **do**
    for $j = i + 1$ fins a $m_0 - 1$ **do**
        Afegir una aresta entre i i j al graf g
    end for
    $connection\_degree[i] = m_0 - 1$
end for
for $i = m_0$ fins a $n - 1$ **do**
    Inicialitzar un vector buit *candidates*
    while la longitud de *candidates* és menor que m **do**
        $selectedNode = \text{preferentialAttachment}(connection\_degree)$
        if $selectedNode$ no està en *candidates* **then**
            Afegir $selectedNode$ a *candidates*
        end if
    end while
    for cada j en *candidates* **do**
        Afegir una aresta entre i i j al graf g
        $connection\_degree[j] += 1$
    end for
    $connection\_degree[i] = m$
end for
Retornar el graf g

El generador de grafs de Barabási-Albert crea un graf que segueix el model de creixement de xarxes, on s'afegeixen nodes nous que es connecten a nodes existents en funció del seu grau de connexió. Comença amb un conjunt inicial de m_0 nodes completament connectats, i cada nou node que s'afegeix selecciona m nodes existents per connectar-se, amb una probabilitat proporcional al seu grau de connexió.

Aquesta selecció es realitza mitjançant la funció `preferentialAttachment`, que s'encarrega de seleccionar un node existent basant-se en el seu grau de connexió. La funció funciona de la següent manera:

Algorisme 5 Preferential Attachment

Entrada: *connection_degree* (vector de graus de connexió)
Sortida: *chosen* (node seleccionat)

```
Inicialitzar degree_sum = 0, temp_sum = 0
for cada i en connection_degree do
    degree_sum += i
end for
random_num = rand() mod degree_sum
for i = 0 fins a length(connection_degree) - 1 do
    temp_sum += connection_degree[i]
    if random_num ≤ temp_sum then
        chosen ← i

    end if
end for
Retornar chosen
```

Aquesta funció calcula la suma total dels graus de connexió de tots els nodes existents i selecciona aleatòriament un node, on la probabilitat de seleccionar cada node és proporcional al seu grau de connexió. Així, els nodes amb més connexions tenen una major probabilitat de ser seleccionats, promovent el creixement de xarxes amb característiques d'escala.

4 Algoritmes

En aquesta secció es presenten els principals algorismes emprats en l'estudi de la transició de fase. A continuació, detallarem cada algorisme i donarem una breu explicació d'aquests. Els algorismes abordats són els següents:

4.1 Percolació per Arestes

Algorisme 6 Percolació d'Arestes en un Graf

Entrada: Graf $G = (V, E)$ amb n nodes, probabilitat p
Sortida: Graf percolat $G_\mathrm{p} = (V, E')$ on $E' \subseteq E$

Crear un nou graf buit G_p amb n nodes (còpia profunda de G)
for cada node u en V **do**
    for cada node v en la llista d'adjacència d'u d'el graf G **do**
        if $u < v \ \wedge \ rand01() > p$ **then**
            Afegir l'aresta (u, v) al graf G_p
        end if
    end for
end for
Retornar el graf G_p

Aquest algorisme retorna el graf percolat a partir del graf original. Això s'aconsegueix eliminant les arestes amb una probabilitat p. Per evitar processar una mateixa aresta més d'una vegada, s'utilitza la condició $u < v$, ja que en grafs no dirigits una aresta (u, v) és equivalent a (v, u). Tal com s'ha mencionat prèviament, la funció `rand01()` genera un nombre aleatori entre 0 i 1, que es compara amb la probabilitat p per decidir si es manté o s'elimina una aresta.

4.2 Percolació per Nodes

Algorisme 7 Percolació de Nodes en un Graf

Entrada: Graf $G = (V, E)$ amb n nodes, probabilitat p
Sortida: Graf percolat $G_\mathrm{p} = (V', E')$ on $V' \subseteq V$ i $E' \subseteq E$

Inicialitzar un vector *posicioNodes* de longitud n
Assignar $nbNodesVius = n$
for cada node u en V **do**
  Generar un valor aleatori $r = \texttt{rand01}()$
  if $r < p$ **then**
    Marcar el node u com a fallat
    Decrementar $nbNodesVius$
  else
    Actualitzar la posició del node viu $posicioNodes[u] = u - n + nbNodesVius$
  end if
end for

Crear un graf buit G_p amb $nbNodesVius$ nodes
for cada node u en V **do**
  if u no ha fallat **then**
    for cada node v en la llista d'adjacència de u en G **do**
      if v no ha fallat $\wedge$ $u < v$ **then**
        Afegir l'aresta $(posicioNodes[u], posicioNodes[v])$ a G_p
      end if
    end for
  end if
end for
Retornar el graf G_p

Aquest algorisme retorna el graf percolat a partir del graf original eliminant nodes amb una probabilitat p. Per a cada node, es genera un valor aleatori entre 0 i 1 mitjançant la funció `rand01()`. Si aquest valor és inferior a p, el node es considera eliminat (fallat) i no es conservarà en el graf resultant.

Els nodes que sobreviuen són reindexats per assegurar que el nou graf té una numeració consecutiva de nodes. Les arestes només es conserven si ambdós nodes que connecten han sobreviscut, i es manté la condició $u < v$ per evitar afegir la mateixa aresta dues vegades, ja que en els grafs no dirigits una aresta (u, v) és equivalent a (v, u). Així, el resultat és un graf amb una mida reduïda en funció de la probabilitat p, mantenint només els nodes i arestes que han "sobreviscut" al procés de percolació.

4.3 Càlcul de Components Connexes

Algorisme 8 Càlcul del Nombre de Components Connexes

Entrada: Graf $G = (V, E)$ amb n nodes
Sortida: Nombre de components connexos $componentCount$

Inicialitzar un vector *visited* de longitud n amb valors $false$
Inicialitzar $componentCount = 0$
for cada node $i = 0$ fins a $n - 1$ **do**
  if el node i no ha estat visitat **then**
    Incrementar $componentCount$
    Realitzar una DFS a partir del node i, marcant els nodes visitats
  end if
end for
Retornar $componentCount$

L'algorisme utilitza una cerca en profunditat (DFS) per explorar cada component connex del graf. Cada vegada que es troba un node no visitat, es crida la funció `dfs` per explorar recursivament tots els nodes connectats a aquest node. Aquesta crida recursiva assegura que tots els nodes del mateix component quedin marcats com a visitats, evitant comptar-los més d'una vegada.

5 Experimentació

5.1 Metodologia

Per dur a terme l'experimentació del projecte, hem utilitzat diferents eines. Hem programat dos programes en `C++`, un llenguatge que ens ofereix molta eficàcia temporal i espacial. Aquests programes són el `main` i el `runner`. També hem dissenyat un fitxer de classe `graph` amb tots els atributs i funcions necessàries per operar amb els grafs. Aquesta classe representa els grafs com a llistes d'adjacència.

Per compilar aquests programes, hem fet ús del programari lliure `make`, que automatitza i paral·litza el compilatge i l'enllaç.

A més, hem dissenyat scripts per a l'interpret `R`, que és un programari de tractament de dades que ens analitzarà i generarà gràfics dels resultats dels estudis, que estaran en format `.csv`.

Més informació del procés d'experimentació es pot trobar en el GitHub del projecte, premeu `aquí` per accedir-hi. Allà, a part del codi, també podreu consultar més informació sobre la generació de grafs, les dependències del programa per compilar-lo i executar-lo, com inserir els paràmetres pels programes i més.

El programa `main`, mitjançant la classe `graph`, ens ha permès analitzar les propietats del canvi de fase a partir dels paràmetres inicials. Aquests paràmetres són els següents:

- **RandomSeed**: La llavor per al generador aleatori.
- **NúmeroMínimNodes**: El nombre mínim de nodes del graf.
- **NúmeroMàximNodes**: El nombre màxim de nodes del graf.
- **NúmeroNodesStep**: Increment dels nodes en cada iteració.
- **IteracionsPerObtenirResultat**: El nombre de vegades que es provarà la configuració per probabilitat p de percolació i per nombre de vèrtex n.
- **ModePercolació**: Tipus de percolació per nodes o per arestes.
- **PathResultat**: Fitxer on es guardaran els resultats.
- **AlgorismeGeneradorGraf**: Algoritme utilitzat per generar el graf (per exemple, Erdős-Rényi, Square-Grid, etc.).
- **ParàmetresAlgorisme**: Paràmetres addicionals per al generador de graf (opcional segons l'algorisme).

A partir d'aquests paràmetres, el programa `main` escriurà un fitxer `PATH.csv` que posteriorment serà analitzat mitjançant el software de tractament de dades `R`.

Per altra banda, tenim el programa `runner`, que rebrà com a input un fitxer de text. Aquest fitxer tindrà un llistat de paràmetres per diferents experiments del programa `main`. Un exemple d'això seria:

```
RGN   MIN MAX  STEP ITs   PERC-MODE  RESULT-PATH       GEN-ALGORITM     PARAMETERS-GEN
--------------------------------------------------------------------------------------
21312 10  100   10  1000  NODE_PERC  ./data/test1.csv Erdos-Renyi         0.1
35353 50  500   50  1000  EDGE_PERC  ./data/test2.csv Random-Geometric    0.3
72479 100 1000  100 100   EDGE_PERC  ./data/test3.csv Square-Grid
```

El programa `runner`, per cada fila del fitxer que rep, inicialitzarà una instància del programa `main`, aconseguint d'aquesta manera automatitzar molt més els tests, podent córrer diferents programes `main` simultàniament.

5.1.1 Programa Main

Per entendre els resultats també s'ha d'entendre les decisions que s'han pres per la recollida de dades. Analitzarem el programa `main` mitjançant un pseudocodi per no entrar en conceptes avançats de `C++`. A continuació vegeu una mostra del pseudocodi:

Algorisme 9 Descripció de l'experiment

```
Seleccionar opcions de configuració
Inicialitzar el generador de nombres aleatoris
Obrir l'arxiu CSV i escriure la capçalera
for n in range(MIN_NB_NODES, MAX_NB_NODES + 1, NB_NODES_STEP) do
    for p des de 0 fins a 1 amb pas 0.01 do
        Inicialitzar el comptador de grafs connexos
        for i = 0 fins a TRIES_PER_P do
            repeat
                Generar el graf seleccionat(p, n, params)
            until el graf és connex
            Aplicar percolació (per nodes o arestes) al graf
            if el graf percolat és connex then
                Incrementar el comptador de grafs connexos
            end if
        end for
        Escriure entrada resultant al CSV
    end for
end for
```

Ara, analitzarem el codi. Per començar, el programa preguntarà per totes les opcions necessàries. D'aquesta manera, ens podem permetre tenir un sol programa que pugui fer tot el que necessitem i que sigui altament modular. S'utilitzarà la llavor per generar nombres aleatoris, i així l'experiment podrà ser repetit amb els mateixos resultats. Després, crearà el fitxer `PATH.csv`, al qual s'hi inseriran entrades que posteriorment s'analitzaran.

Ara analitzarem l'algorisme encarregat de generar els resultats. Vegeu com primerament iterarem sobre n tantes vegades com s'hagi indicat en la entrada. Alhora, també iterarem per cada n sobre una probabilitat de fallida de percolació. Aquest bucle tindrà 100 iteracions, $p \in \{0.00, 0.01, \ldots, 1.00\}$. A més d'aquests dos bucles, iterarem una altra vegada sobre p i n tantes vegades com l'usuari hagi indicat en l'apartat *IteracionsPerObtenirResultat*. Així, es farà una mitjana amb més o menys mostres.

S'iniciarà un comptador a 0 que representarà el nombre de grafs percolats connexos. S'utilitzarà el generador de grafs seleccionat a les opcions per generar el graf que posteriorment serà percolat. Vegeu que aquí generarem grafs fins a aconseguir un graf connex. Aquesta decisió la vam prendre per tenir una representació més acurada de la transició de fase. Més endavant, es tornarà a considerar aquesta decisió, ja que hi ha grafs, com ara el *Random Geometric Graph*, que requereixen un paràmetre r, el qual, amb valors petits de r, acostuma a generar grafs no connexos.

Per acabar, percolarem el graf $G(V, E)$ de la manera que s'hagi especificat a l'input, ja sigui per nodes o per arestes, obtenint G_{p}. A G_{p} se li aplicarà un algorisme que determinarà si el graf és connex. Si ho és, s'incrementarà el comptador. Quan les *IteracionsPerObtenirResultat* s'hagin completat, s'escriurà l'entrada resultant al fitxer `PATH.csv`.

5.2 Lectura de taules i gràfics

Abans de procedir amb els experiments, es presenten les claus per interpretar correctament les taules i gràfics que acompanyen els resultats.

Gràfics: Els gràfics mostren l'evolució dels experiments en funció del paràmetre p (probabilitat de fallida) a l'eix de les X, mentre que l'eix Y mostra la probabilitat que el graf percolat G_{p} sigui connex. A més, en cada gràfic col·locarem més d'un experiment, donades diferents valors de n (nombre de nodes), pintant de colors diferents cada experiment. Aquest últim paràmetre a vegades canviarà, ja que en grafos, com per exemple en el quadrat, mesurarem amb el paràmetre m, que és el nombre de files.

Taules: Les taules contenen les mètriques que són més difícils de veure amb els gràfics. Cada fila representa una instància específica de l'experiment, mentre que les columnes mostren:

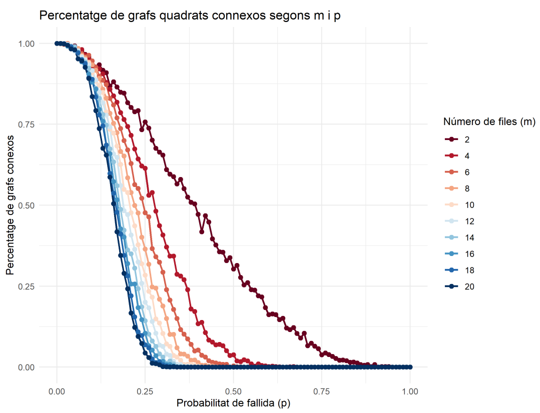
- **N**: nombre de nodes de l'experiment (pot variar depenent del graf).
- ***Threshold***: punt d'inflexió en què apareix la transició de fase.
- **Probabilitat Connex**: la probabilitat que el graf percolat sigui connex.
- **Taxa de Canvi**: pendent del segment en el punt *threshold*.

Creiem convenient explicar com calculem el punt d'inflexió o punt *threshold*. El trobarem com el segment amb el pendent més pronunciada.

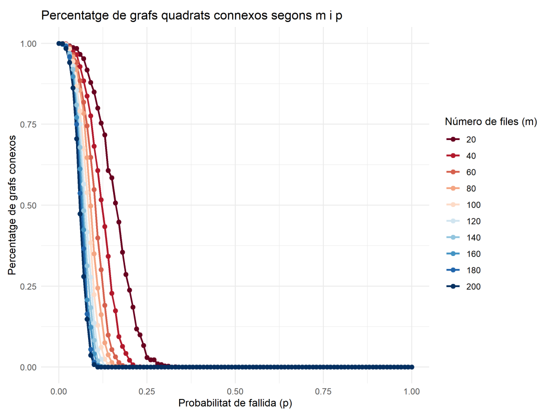
5.3 Graella quadrada

Per estudiar la possible transició de fase a graelles quadrades hem decidit fer proves amb grafs de mida petita, des de 2x2 nodes a 20x20, així com amb grafs de mida més gran, de 20x20 nodes a 200x200. Això ens servirà per estudiar el comportament de la percolació tant per node com per aresta en diferents escales de complexitat. Hem triat aquestes mides de grafs per poder comparar resultats en diferents dimensions i observar com la mida del graf influeix en la percolació, formació de components connexes i transició de fase.

5.3.1 Percolació per arestes



4321 2 20 2 1000 EDGE_PERC
./data/squareE-2-20-1.csv Square-Grid



4321 2 20 2 1000 EDGE_PERC
./data/squareE-2-20-1.csv Square-Grid

Figura 1: Comparativa de percolació per arestes del grafs quadrats amb mides diferents

La *Figura 1* mostra una comparació de la percolació per arestes en grafs quadrats de mides diferents. Ambdós gràfics representen la relació entre la probabilitat de fallida p (eix x) i el percentatge de grafs connexos (eix y). A mesura que la p augmenta, més arestes s'eliminen i per tant hi ha menys connectivitat en els grafs. El gràfic de l'esquerra mostra els resultats per grafs de mida petita, de 2 a 20 files, mentres que el de la dreta representa els resultats per grafs més grans, de 20 a 200 files.

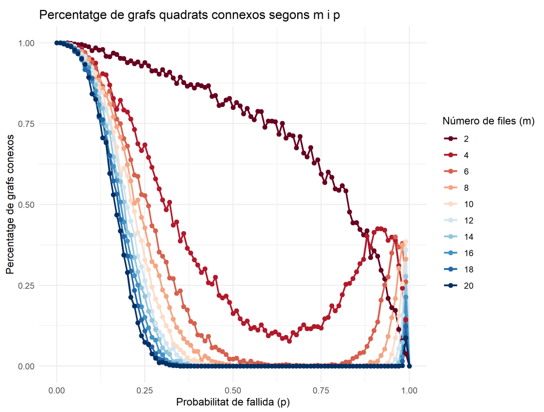
En els dos gràfics podem veure com a mesura que els grafs són més grans la pèrdua de connectivitat comença a valors més petits de p. En canvi, els grafs més petits com els de 2 o 4 files continuen connectats durant més temps, arribant a valors més grans de p abans de perdre completament la connectivitat. A més a més, en grafs grans es pot observar una baixada molt més brusca en el percentatge de grafs connexos que ens indica una transició de fase més marcada.

M	Threshold p	Percentatge Connex	Tasa de Canvi
2	0.24	0.73	-5.90
4	0.26	0.53	-8.30
6	0.27	0.37	-9.80
8	0.24	0.40	-7.50
10	0.19	0.55	-7.70
12	0.18	0.56	-8.70
14	0.17	0.56	-7.40
16	0.18	0.43	-9.00
18	0.15	0.60	-8.80
20	0.17	0.42	-8.70
40	0.15	0.23	-11.40
60	0.11	0.40	-14.90
80	0.09	0.50	-14.90
100	0.10	0.22	-16.00
120	0.07	0.56	-18.50
140	0.07	0.48	-19.60
160	0.08	0.21	-21.60
180	0.06	0.54	-21.30
200	0.06	0.47	-23.20

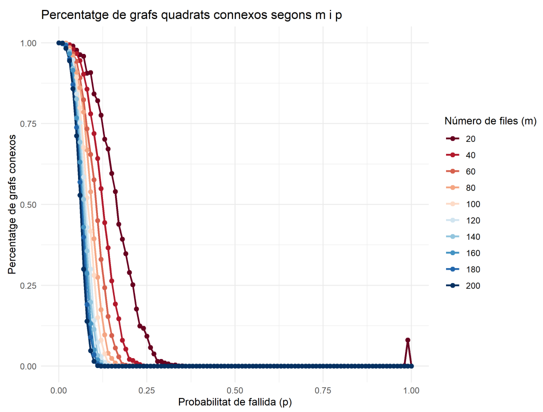
Taula 1: Dades de connectivitat de grafs quadrats

Amb aquesta comparació podem veure clarament com la mida del graf influeix en la rapidesa de la pèrdua de connectivitat i el valor de p on hi ha la transició de fase. En grafs petits la transició és més gradual i el valor de probabilitat de fallida que provoca la desconnexió completa és més alt. Per contra, en els grafs grans la pèrdua de connectivitat és molt més abrupta i el valor de p que provoca la desconexió total del graf és més petit.

5.3.2 Percolació per nodes



5202 2 20 2 1000 NODE_PERC ./data/squareN-2-20-1.csv Square-Grid



4657 20 200 20 1000 NODE_PERC ./data/squareN-20-200.csv Square-Grid

Figura 2: Comparativa de percolació per nodes del grafs quadrats amb mides diferents

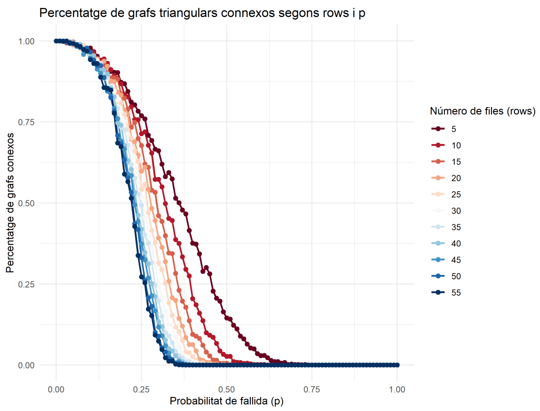
La *Figura 2* mostra la comparació de la percolació per nodes en grafs quadrats de mides diferents. Com en la percolació per arestes, el gràfic de l'esquerra mostra els resultats per grafs petits i el de la dreta grafs més grans. Els eixos són també els mateixos, i cada linea del graf mostra la relació entre la probabilitat de fallida p i el percentatge de grafs connexos segons la mida del graf.

5.4 Graella triangular

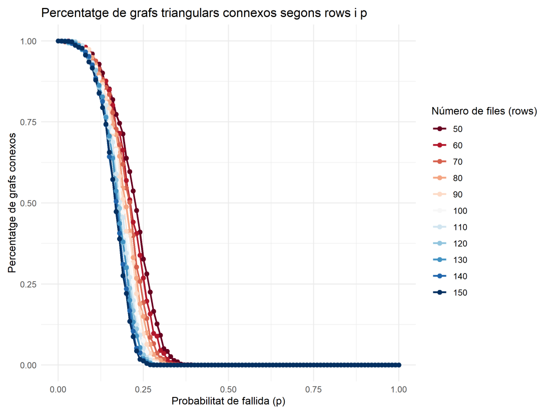
Ara estudiem la transició de fase en graelles triangulars. En aquest cas només havíem d'estudiar la percolació per aresta i ho hem fet com en les graelles quadrades, amb grafs de mida petita i mida més gran. En aquest cas, els grafs petits comencen amb 5 files fins a 55 files, amb increment de 5 files cada vegada, mentre que la mida gran son grafs de 50 a 150 files amb increments de 10 files.

Hem decidit fer la percolació per aresta perquè volíem veure si el fet de tenir una graella similar a la graella quadrada, pero amb els vertex del mig (el que no estan a la vora del graf i per tant tenen totes les arestes possibles) de grau més alt influiria en el valor de la probabilitat de fallida on hi ha la transició de fase. Creiem que en aquest cas la probabilitat de fallida on hi ha la transició de fase serà un valor més alt que en graelles quadrades ja que el grau dels vèrtex és més alt i per tant hi ha més arestes que han de fallar per desconnectar els nodes. Després de fer aquesta experimentació, hem decidit fer també l'estudi per percolació de node per poder comparar els resultats amb els obtinguts en graelles quadrades.

5.4.1 Percolació per aresta



3497 5 55 5 1000 EDGE_PERC
./data/triangularE-5-55-1000.csv Triangular-Grid



3363 50 150 10 1000 EDGE_PERC
./data/triangularE-50-150.csv Triangular-Grid

Figura 3: Comparativa de percolació per arestes de graelles triangulars amb mides diferents

La *Figura 3* mostra la comparació de percolació per arestes de graelles triangulars amb els resultats dels grafs petits a l'esquerra i els grans al gràfic de la dreta. La informació que cad agràfic proporciona és la mateixa que en l'experimentació de les graelles quadrades, cada línea indica la relació entre percentatge de grafs connexos i probabilitat de fallida p en funció del número de files.

En els dos casos podem observar que hi ha una transició de fase ja que hi ha una caiguda brusca en un valor determinat de p, però en el cas dels grafs petits aquesta caiguda és més gradual, la transició de fase no està tant marcada. En canvi, en grafs amb més files la transició és més ràpida i si ens fixem en el valor de la probabilitat de fallida on arribem a la desconnexió total, la p que marca la transició de fase disminueix a mesura que el graf és més gran.

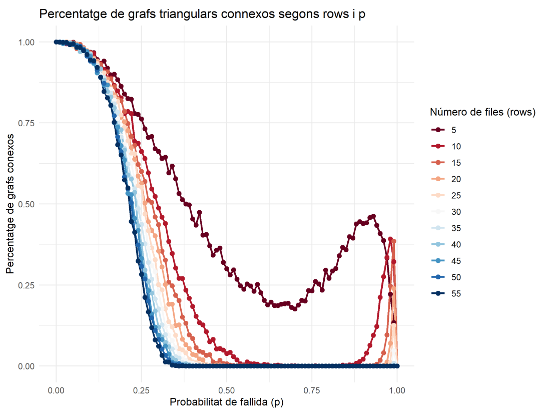
Per tant, igual que en els grafs quadrats la mida del graf afecta el valor de la probabilitat de fallida on ocorre la transició de fase, la qual disminueix a mesura que el graf augmenta de mida. Pel que fa a la comparació que ens interessava amb les graelles quadrades, tot i que observant els gràfics ja es pot veure que la p on hi ha la desconnexió total és més petita en les graelles quadrades, en aquesta taula podem observar resultats de grafs amb el mateix número de files per poder veure millor la comparació.

Amb aquesta comparació, podem veure que grafs amb el mateix número de files tenen valors de p on hi ha la transició de fase molt més petits, normalment casi la meitat, en les graelles quadrades.

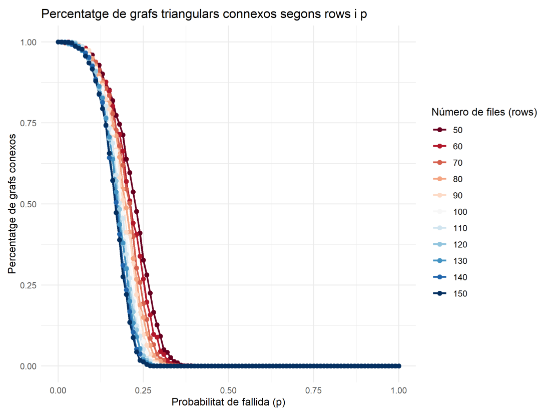
Número files	Valor p en graella quadrada	Valor p en graella triangular
10	0.46	0.62
20	0.34	0.52
40	0.24	0.40
60	0.20	0.38
80	0.18	0.33
100	0.16	0.31
120	0.14	0.30
140	0.13	0.28

Taula 2: Dades de transició de fase en graelles

5.4.2 Percolació per node



7002 5 55 5 1000 NODE_PERC
./data/triangularN-5-55-1000.csv Triangular-Grid



4056 50 150 10 1000 NODE_PERC
./data/triangularN-50-150.csv Triangular-Grid

Figura 4: Comparativa de percolació per nodes de graelles triangulars amb mides diferents

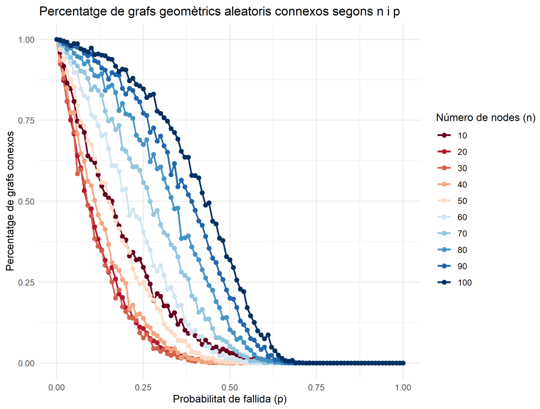
5.5 Graf geomètric aleatori

L'estudi de transició sobre grafs geometrics aleatoris depèn, a més de la probabilitat de percolació p, dels dos paràmetres que s'utilitzen per a la creació del graf: el nombre de nodes n i el radi de connectivitat r. En aquesta experimentació, s'han fet 6 casos de prova, on es compara l'impacte de tots dos paràmetres en diferents nivells. En el cas del nombre de nodes n, els valors que s'han utilitzar són de 10 a 100 i de 100 a 1000, donant-nos resultats per a grafs amb pocs nodes fins a grafs amb una quantitat relativament gran de nodes. Per part del radi de connectivitat r, els valors que s'han escollit són de 0.2, 0.5, 0.8, que ens permeten observar grafs amb una connectivitat bastant feble en el cas del primer valor, grafs amb una connectivitat molt alta amb $r = 0.8$ i un mig entre tots dos valors.

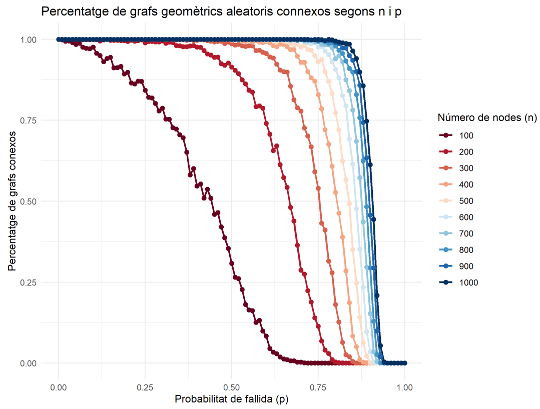
5.5.1 Percolació per arestes

En aquesta primera comparativa veiem com grafs amb una connectivitat feble amb pocs nodes deixen de generar grafs connexos quan hi ha una probabilitat de percolació relativament baixa. No obstant, a mesura que augmenten els nodes, la quantitat de grafs connexos en funció de la probabilitat de percolació comença a creixer de manera notable (imatge 2). Això ens mostra que, tot i ser grafs amb una connectivitat teòrica feble, amb una quantitat de nodes suficientment gran es pot aconseguir una transició de fase tot i que la probabilitat de percolació sigui alta.

En aquesta segona comparativa, les imatges mostres mesures sobre grafs amb una connectivitat teòrica major als anteriors. L'impacte d'aquesta connectivitat major es veu reflexat clarament en tots dos experients. En els casos de grafs amb una quantitat de nodes no gaire elevada, el percentatge de grafs connexos en funció de la probabilitat de percolació és bastant elevat fins i tot per al cas de grafs d'únicament 10 nodes (imatge 1). Més enllà d'aquesta quantitat, el percentatge incrementa de forma notòria fins a arribar un punt en el que la diferència entre les probabilitats de percolació per a cada quantitat de nodes és irrisoria.

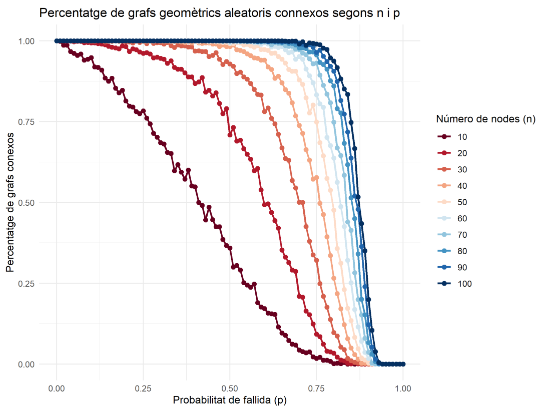


9283948 10 100 10 1000 EDGE_PERC
./data/rgg10.csv Random-Geometric 0.2

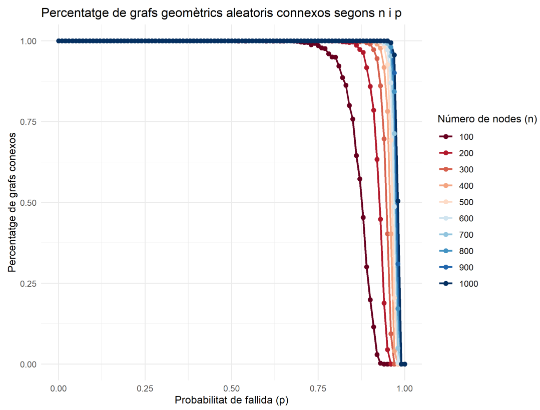


9283948 10 100 10 1000 EDGE_PERC
./data/rgg16.csv Random-Geometric 0.2

Figura 5: Comparativa de percolació per arestes del grafs geometrics aleatoris amb n = 10..100 (1) i n = 100..1000 (2), r = 0.2



2389476 10 100 10 1000 EDGE_PERC
./data/rgg11.csv Random-Geometric 0.5



6673428 100 1000 100 1000 EDGE_PERC
./data/rgg17.csv Random-Geometric 0.5

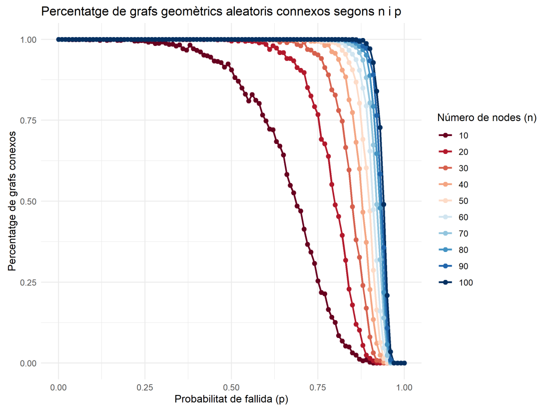
Figura 6: Comparativa de percolació per arestes del grafs geometrics aleatoris amb amb n = 10..100 (1) i n = 100..1000 (2), r = 0.5

Per últim, en aquesta comparativa s'utilitzen grafs geometrics aleatoris amb un radi de connectivitat molt elevat, donant-lis una forma seblant a un graf complet. Per aquest motiu, fins que la probabilitat de percolació no és molt alta, els grafs generats segueixen sent connexos una vegada aplicada la transició de fase. Evidentment, l'augment de nodes provoca que el percentatge de grafs connexos creixi ja que els nodes tenens més probabilitats d'estar units a altres nodes (imatge 2), fet que ha succeït a totes les comparatives.

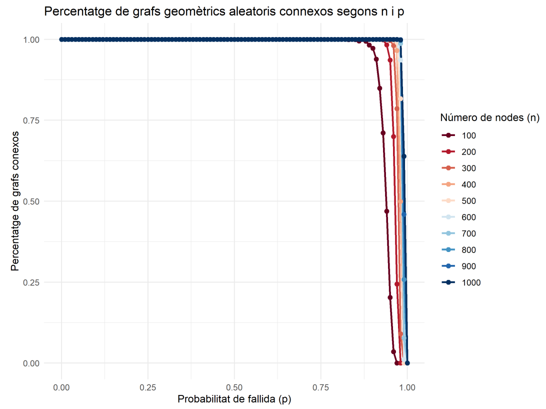
5.6 Graf de Barabási-Albert

L'estudia que s'ha realitzat sobre els grafs generats amb el model de Barabási-Albert és un de transició de fase sota percolació per nodes. El motiu és que es vol veure l'impacte d'una percolació a un graf que conté molts vertexs units entre sí per un o més "hubs" centrals, l'impacte que tindria la fallida d'algun d'aquests nodes o, per altra banda, el no impacte que pot tenir sobre nodes que tenen una estructura de fulla, és a dir, nodes que estan connectats unicament a un altre node i pertant la seva fallida segueix permetent un graf connex.

Els experiments realitzats sobre aquests grafs depenen de tres paràmetres: nombre de nodes n, nombre no nodes inicials o "hubs" $m0$, i el nombre d'arestes que han de tenir com a mínim cada node que no és "hub". Per als nostres experiments, hem decidit donar els següents valors: per al nombre de nodes n = 10..100, n = 50..500, per al nombre inicial de nodes o "hubs" $m0$ = 2, $m0$ = 5, $m0$ = 10, i per el nombre d'arestes incidents a la resta de nodes m = 1, m = 2, m = 5. Abans de veure els resultats, hem cregut adient une explicació del perqué s'han agafat aquests valors i no d'altres. Els grafs construïts amb



8273563 10 100 10 1000 EDGE_PERC
./data/rgg12.csv Random-Geometric 0.8

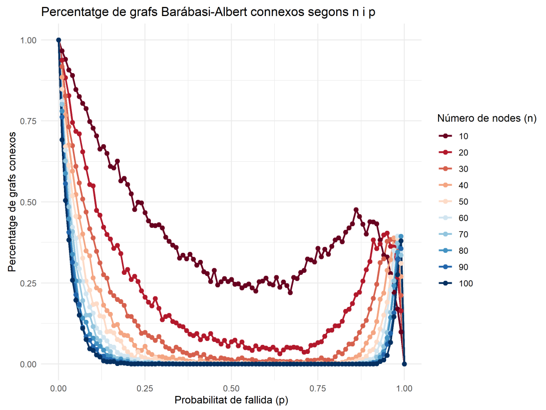


9293745 100 1000 100 1000 EDGE_PERC
./data/rgg18.csv Random-Geometric 0.8

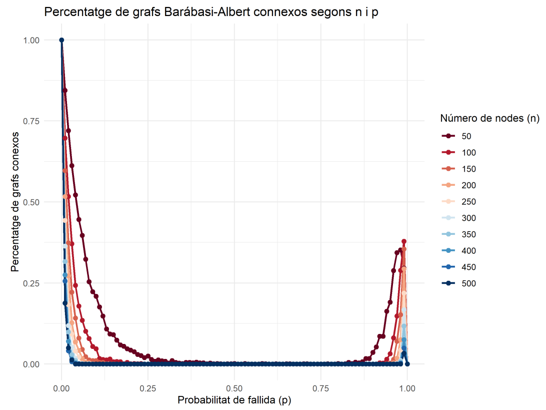
Figura 7: Comparativa de percolació per arestes del grafs geometrics aleatoris amb n = 10..100 (1) i n = 100..1000 (2), $r = 0.8$

el model de Barabási-Albert són típicament utilitzats com a xarxes on els nodes s'uneixen mitjançant "hubs". Per aquest motiu, hem decidit utilitzar aquests valors ja que representen d'una manera bastant realista una xarxa generada per aquest model d'una manera computacionalment viable.

5.6.1 Percolació per nodes

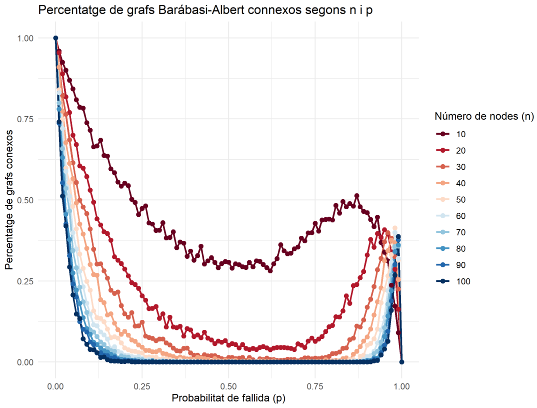


4575745 10 100 10 1000 NODE_PERC ./data/ba9.csv
Barabasi-Albert 2 1

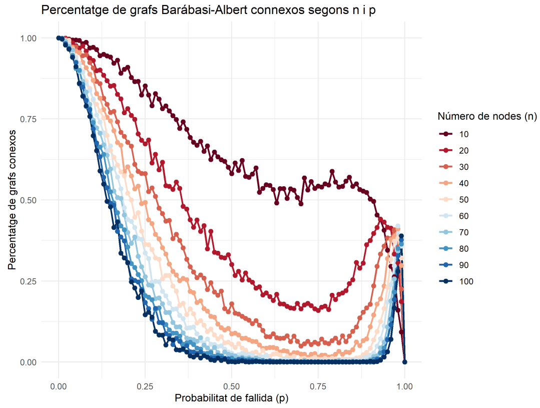


7673658 50 500 50 1000 NODE_PERC ./data/ba13.csv
Barabasi-Albert 2 1

Figura 8: Comparativa de percolació per arestes del grafs Barabási-Albert amb n = 10..100 (1) i n = 50..500 (2), $m0 = 2$, $m = 1$

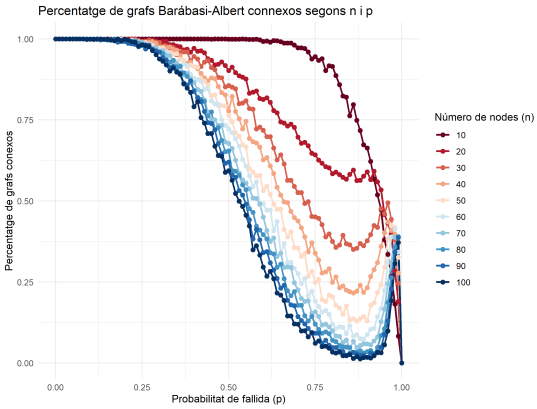


9937843 10 100 10 1000 NODE_PERC ./data/ba10.csv
Barabasi-Albert 5 1

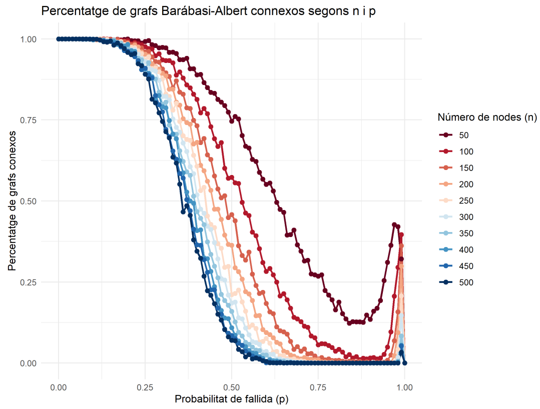


8800547 10 100 10 1000 NODE_PERC ./data/ba11.csv
Barabasi-Albert 5 2

Figura 9: Comparativa de percolació per arestes del grafs Barabási-Albert amb n = 10..100, $m0$ = 5, m = 1 (1) i m = 2 (2)



6657323 10 100 10 1000 NODE_PERC ./data/ba15.csv
Barabasi-Albert 10 5



9872345 50 500 50 1000 NODE_PERC ./data/ba16.csv
Barabasi-Albert 10 5

Figura 10: Comparativa de percolació per arestes del grafs Barabási-Albert amb n = 10..100 (1) i n = 50..500 (2), $m0$ = 10 m = 5

6 Conclusions

7 Bibliografia

- Wikipedia. *Model d'Erdős-Rényi*.

8 Annex