ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

Estructures de Dades 2019-2020

Pràctica 2. Estructures de dades avançades

Introducció i context

En aquesta pràctica haureu d'implementar dues estructures de dades avançades (grafs i heaps) mitjançant el llenguatge de programació Java, que a continuació s'utilitzaran per tal de solucionar un problema real dins el camp de les xarxes complexes.

Les xarxes [1] (grafs) són àmpliament usades per a modelar sistemes composats per un conjunt d'unitats que interactuen entre sí. És fàcil trobar exemples en la nostra vida diària: xarxes socials, tant virtuals (Twitter, Facebook, WhatsApp o correu electrònic) com físiques (relacions amb els nostres amics o família), xarxes de transport (carreteres, metro, bus, vaixells o aeroports), xarxes de distribució elèctrica, o xarxes de relacions comercials entre països. En aquesta pràctica, mitjançant una sèrie de passos, intentarem veure com podem avaluar la robustesa d'aquestes xarxes en front d'atacs dirigits.

Imaginem que tenim la xarxa d'aeroports, on cada node és un aeroport, i cada aresta de la xarxa conté informació del volum de passatgers que viatgen entre els dos aeroports. Si es produeix una disrupció de funcionament en un aeroport, per exemple una vaga dels controladors aeris, un accident que afecta les pistes, o un atemptat terrorista, les línies que el connecten deixen de funcionar temporalment. Si es tracta d'un aeroport important, no només queda afectada la ciutat corresponent, sinó tota la xarxa en el seu conjunt.

Dins el camp dels sistemes complexos, una de les formes d'analitzar la robustesa de xarxes consisteix en realitzar una anàlisi de percolació [2]. La idea és que, conforme es van eliminant nodes o enllaços, la xarxa comença a fragmentar-se. Una xarxa és robusta si manté una bona connectivitat tot i la pèrdua de nodes o enllaços. Per a mesurar la connectivitat s'utilitza el concepte de components connexes d'un graf, que es veu a teoria [3].

Estructura de dades a implementar 1: Graf

Estructura de dades Graf

Es demana implementar en Java les estructures necessàries que permetin emmagatzemar informació en format de graf no dirigit i pesat, així com les operacions que es detallen a en aquest apartat. L'estructura de graf ha de ser genèrica, permetent emmagatzemar qualsevol tipus

d'informació als nodes i als enllaços. Per exemple, en la xarxa d'aeroports, ens pot interessar guardar un identificador i una posició geogràfica al node, i el volum de passatgers a l'enllaç, mentre que en altres xarxes la informació pot ser molt diferent. Es recomana realitzar una implementació basada en llistes d'adjacència, però cada alumne és lliure d'escollir o dissenyar el tipus d'implementació que vulgui; això sí, les matrius d'adjacència no estan permeses per a guardar el graf ni els pesos, ja que són extraordinàriament ineficients per a resoldre el nostre problema de percolació.

La qualitat del disseny es valorarà a la nota final de la pràctica. Fixeu-vos que en aquesta pràctica ja no estem donant cap llista d'operacions a implementar; vosaltres mateixos, a través de les necessitats que observeu, podeu escollir quines operacions ha d'oferir l'estructura per realitzar el procés de percolació.

Funcionalitats generals a implementar

- Llegir una xarxa format Pajek [4,5], que és un tipus d'arxiu de text que permet emmagatzemar l'estructura de la xarxa, així com els atributs als nodes i a les arestes. La xarxa es suposa no dirigida.
- Detecció de components connexes utilitzant un algorisme d'exploració com els explicats a classe.
- Anàlisi de percolació. Visualització de l'evolució del nombre de components connexes de la xarxa, i de la mida de les dues components connexes més grans, a mesura que anem extirpant nodes de la xarxa.

Percolació

Tal com s'ha comentat abans, l'anàlisi de percolació d'una xarxa ens permet mesurar quina robustesa té en front a la pèrdua d'operativitat dels components del sistema. En aquest procés (iteratiu) analitzarem com la xarxa es va desintegrant en components connexes cada vegada més petites a mesura que els nodes s'extirpen de la xarxa. En interessa avaluar la quantitat de components (quants en tenim) i la seva mida. El procediment en termes generals és el següent:

1. Extirpar un node (o un conjunt petit de nodes).
2. Mesurar la funcionalitat de la xarxa: nombre i mida de les components connexes.
3. Repetir 1 i 2 fins que no quedin nodes a la xarxa.

Extirpació d'un node

L'extirpació d'un node es pot considerar de dues formes: eliminar el node de la xarxa, o eliminar només tots els enllaços del node (sense eliminar el node de la xarxa). Podeu seleccionar qualsevol de les dues opcions.

Selecció del node a extirpar

Considerarem dues metodologies d'extirpació obligatòries i una d'opcional:

- *Aleatòria*: selecció d'un node aleatori d'entre els que encara estan actius.
- Per *grau* del node: seleccionar el node de més grau entre els que encara estan actius, on el grau d'un node és el nombre d'enllaços que té.
- Per *strength* del node (opcional): seleccionar el node de més strength entre els que encara estan actius, on strength d'un node és la suma dels pesos dels seus enllaços.

En l'extirpació aleatòria, donat que és no determinista, els resultats poden variar segons la seqüencia de nombres aleatoris que s'hagi generat. Per tal d'obtenir resultats significatius, el procés de percolació s'ha de repetir diverses vegades, i calcular les mitjanes les variables d'interès, explicades en el següent apartat.

En la percolació per grau, els nodes s'aniran extirpant segons el grau, eliminant sempre el node de grau més gran d'entre els que encara romanen a la xarxa. Donat que el procés es determinista, no serà necessari realitzar vàries execucions de l'experiment. La percolació per strength és equivalent però considerant la suma dels pesos de les arestes. Aquests dos tipus de percolació es consideren *atacs dirigits*, en el sentit que s'intenta fer el major mal possible a la xarxa, és dir, fer-la no operativa amb el mínim nombre d'atacs.

Mesures de la funcionalitat de la xarxa

Un dels efectes més importants de la pèrdua de nodes de la xarxa és la seva disgregació en components no connexes. Això significa, per exemple, que en la xarxa d'aeroports es fa impossible viatjar entre certes parelles d'origen-destí, encara que estiguem disposats a fer molts transbordaments.

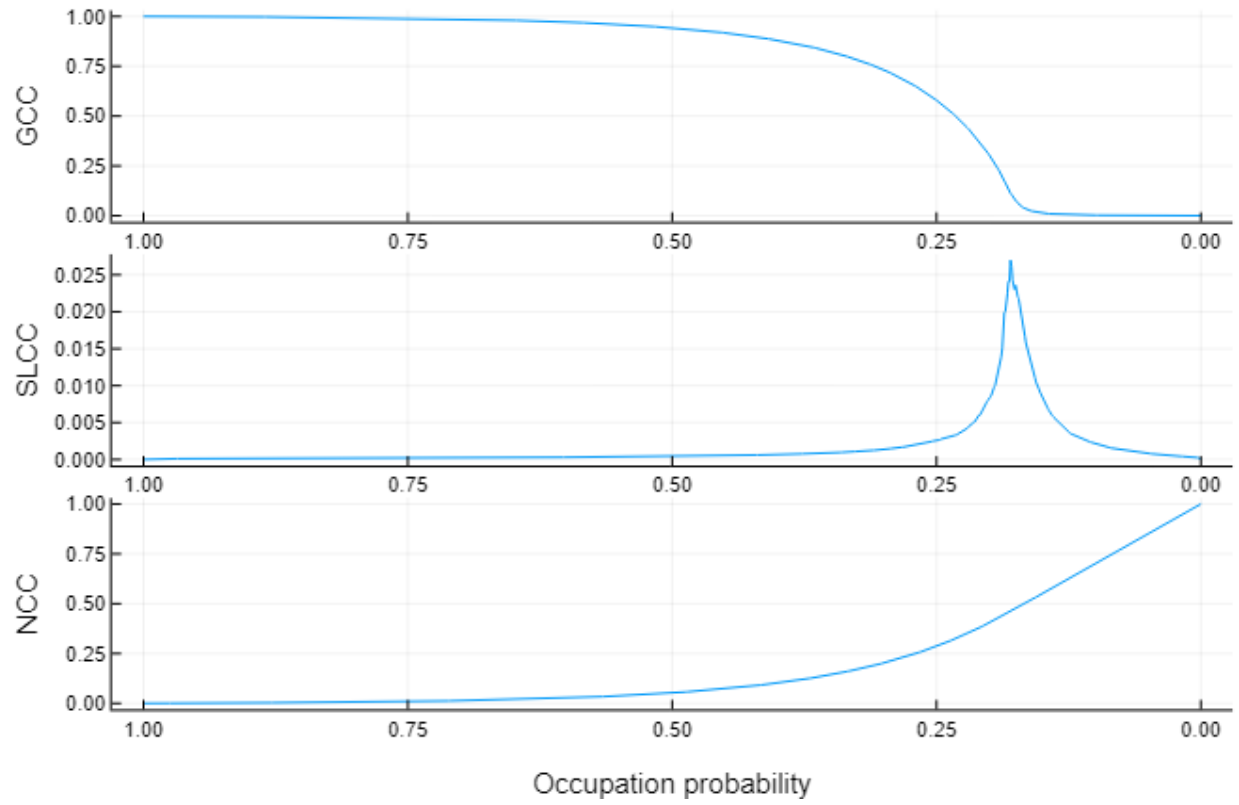
Els paràmetres importants en percolació són els següents:

- *Occupation probability* (OP): Fracció de nodes que no s'han extirpat de la xarxa.
- Nombre de components connexes (NCC): si heu optat per eliminar els nodes de la xarxa en lloc de només els seus enllaços, al nombre de components connexes actual cal que li sumeu el nombre de nodes extirpats, ja que cada node extirpat es considera que forma una component connexa formada per només un node (ell mateix).
- Mida de la component connexa més gran, anomenada la *Giant Connected Component* (GCC).
- Mida de la segona component connexa més gran, anomenada la *Second Largest Connected Component* (SLCC).

Habitualment aquestes mesures (NCC, GCC i SLCC) es presenten normalitzades pel nombre original de nodes de la xarxa, de manera que prenen valors en l'interval [0,1]. La OP, per definició, també es troba dins d'aquest interval.

Procés de percolació

La percolació es fa node a node, seleccionant cada vegada un node segons els tres possibles criteris esmentats anteriorment. Si la xarxa és gran, no cal calcular les components connexes després d'extirpar cada node, es pot fer per exemple cada tres o quatre extraccions. La idea és que, al final del procés, tinguem entre 500 i 1000 conjunts de valors (OP, GCC, SLCC, NCC). El resultat hauria de ser d'aquest estil:



En aquest exemple, amb una xarxa de 10000 nodes, es pot veure com GCC va disminuint a velocitat creixent fins que, aproximadament per OP = 0.18, canvia de tendència i es fa plana i molt propera a 0. Això s'anomena transició de fase, i es fa més evident mentre més gran sigui la xarxa. Observeu també com SLCC té un pic assenyalant el punt de la transició. La corba de NCC és menys interessant, creixent de forma monòtona i sense canvis de tendència. Vosaltres treballareu amb xarxes més petites, i per tant la transició de fase podria ser menys marcada.

Estructura de dades a implementar 2: Heap

Donat que en la percolació amb atacs dirigits s'ha de seleccionar, a cada pas, el node de màxim grau o strength, és necessari implementar una estructura de dades que, de forma eficient, ens permeti saber quin és el següent node que s'ha d'extirpar. Aquesta estructura és el heap. De fet,

en el nostre cas, es tracta d'un Max-heap ja que la percolació es farà donant prioritat als nodes de major grau o strength.

Un detall important és que, a mesura que es van extirpant nodes del graf, el grau i el strength dels nodes adjacents es va reduint; en particular, els graus dels nodes adjacents del que s'extirpa disminueixen en una unitat, i el seu strength es redueix en el valor del pes de l'enllaç que els uneix. Això significa que no hi ha prou amb crear el Max-heap abans de començar la percolació: cal anar actualitzant el valor (i per tant, la posició) dels nodes dins del heap. Per a fer-ho, hi ha dues opcions: (1) eliminar del heap els nodes que han canviat de valor, balancejar-ho, i tornar-los a introduir amb el valor nou; (2) implementar el mètode "decrease-key".

Dades

Us proporcionem les següents xarxes, totes no dirigides i pesades:

- `wtw2000-sym.net`: World Trade Web any 2000, amb 183 nodes.
- `email_URV-edges_betw.net`: Correu electrònic de la URV, amb 1133 nodes.
- `airports_UW.net`: Transport aeri, amb 3618 nodes.
- `powergrid_USA-edges_betw.net`: Distribució elèctrica a USA, amb 4941 nodes.

Resum de les tasques

Les tasques a realitzar es poden resumir en:

- Implementar en Java una estructura de dades graf, genèrica, per a poder emmagatzemar atributs a nivell de node i d'enllaç.
- Implementar en Java una estructura de dades Max-heap, genèrica, per a poder seleccionar de manera eficient els nodes a eliminar del graf.
- Validar les dues estructures anteriors amb un joc de proves adequat.
- Implementar l'algorisme per a trobar les components connexes del graf, i validar el seu bon funcionament.
- Implementar l'algorisme de percolació de xarxes. Pel cas aleatori, repetir el procés almenys 100 vegades, i calcular les mitjanes aritmètiques dels GCC, SLCC i NCC obtinguts. Per a la percolació d'atacs dirigits segons màxim grau o màxima strength, no cal fer mitjanes.
- Aplicar l'algorisme de percolació a les xarxes proporcionades, guardant els resultats en un arxiu CSV. Per a cada xarxa i procediment, cal calcular com a mínim 500 valors dels paràmetres GCC, SLCC i NCC (excepte per a la xarxa WTW, que només té 183 nodes).
- Fer un informe, indicant:
    - Decisions de disseny.
    - Instruccions d'execució.
    - Resultats de la percolació en forma de gràfics semblants al mostrat anteriorment.
    - Taula amb els temps de càlcul, indicant les característiques de l'entorn d'execució.
    - Interpretació dels resultats.

Lliurament

- La pràctica es pot fer individualment o en grups de dues persones.
- Es fa el lliurament d'un únic arxiu PR2-NOM_COGNOMS.zip que conté:
    - Els arxius de codi font
    - L'informe
    - Els arxius de resultats (CSV).

Referències

[1] https://en.wikipedia.org/wiki/Network_theory
[2] https://www.youtube.com/watch?v=_ztNkmDg0mw
[3] https://en.wikipedia.org/wiki/Component_(graph_theory)
[4] http://www.stats.ox.ac.uk/~snijders/PajekIntro.pdf
[5] https://gephi.org/users/supported-graph-formats/pajek-net-format/