Carnegie Mellon University
15-826 Multimedia Databases & Data Mining
Fall 2019 - C. Faloutsos

Default-project Description: `GraphDetective`

Abstract

Short version: for **Phase 1**, check subsection 4-Phase-1, i.e.: (a) do literature survey; and (b) find the injections in the upcoming *'MysteryDataset - unweighted'*.

1 Introduction - Problem description

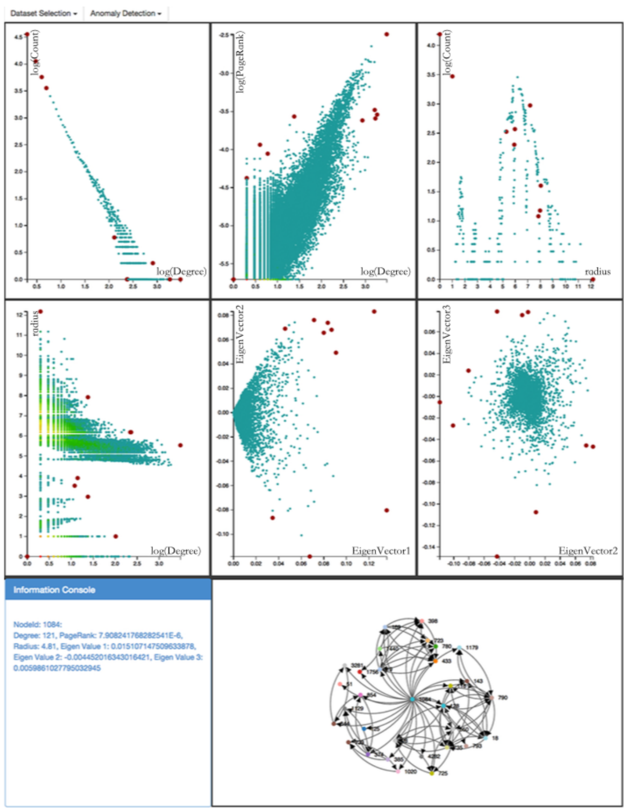
Given a graph (like who-likes-whom in Facebook; or who-reviews-what in Yelp/TripAdvisor), can you find fraudsters, or, at least, suspicious entities?

We have the following goals in this project

1. **Spotting injections**: find suspicious entries, in two graph datasets that we will provide - an unweighted one, and a weighted one. We will inject some strange nodes ('ground truth'), that you will have to discover - hence the project-name 'detective'
2. **Tool development**: develop an interactive graphical user interface, to make the above task easier for you, as well as the multiple colleagues in the industry, academia and government, that have to analyze such graphs. We envision a re-implementation/generalization of the Perseus system [11] (see Figure 1). The goal is to *justify* our responses, and *visualization* provides strong, convincing arguments.
3. **Real Data**: stress-test your system on some publicly available graph datasets, find the most suspicious entities there, and justify your responses. Notice that there will be no 'ground truth' in these datasets, which is often the case in real life.

Motivation: Graphs appear in numerous settings; spotting anomalies and fraudsters is vital. Some settings include:

- *who-friends-whom* on FaceBook. Fraudster may 'buy' friends, from unscrupulous companies, so that they seem more important than they actually are. Similarly, fraudsters may buy 'likes'
- *who-follows-whom* on Twitter. Similarly, fraudsters 'buy' followers, to boost their importance, and the rate they charge for advertizers
- *who-reviews-what* on Yelp, ebay, amazon, tripAdvisor: dishonest sellers may 'buy' fake reviews



NodeId: 1084:
Degree: 121, PageRank: 7.908241768282541E-6,
Radius: 4.81, Eigen Value 1: 0.015107147509633878,
Eigen Value 2: -0.004452016343016421, Eigen Value 3:
0.005986102779503294 5

Figure 1: Screenshot of the 'Perseus' system [11] that we want to generalize in this project. 'Red' dots indicate outliers, that your system should help us investigate further - see bottom panels, with the ego-net on the right.

- *who-calls-whom*: telemarketers in phone networks, would probably have different behavior than normal users
- *fake-news*: fraudsters re-tweeting fake news, would probably form dense subgraphs (all retweeting each others tweets, so that they all look important)
- *health-insurance* fraud: groups of fraudulent doctors, submit similar diagnoses (and expenses), for too many patients.
- *human trafficking* detection: escort-service advertisements, look too similar to each other, if they come from organized crime.

2 Data

Phase 1 : we will provide a real dataset, directed, unweighted dataset, with injections (*'Mystery-Dataset - unweighted'*). It will similar to the 'patent' dataset of HW1, ie, CSV file with `(source, destination)` pairs.

Phase 2 we will provide a *weighted*, directed dataset with injections, (*'MysteryDataset - weighted'*). It will be a CSV file of triplets: `(source, destination, weight)`.

Phase 3 : For stress-testing your system, you may choose datasets among these excellent repositories, with several overlapping datasets.

- `http://konect.uni-koblenz.de/networks/` from the KONECT project.
- `http://snap.stanford.edu/data/index.html` from the SNAP project.
- `http://www-personal.umich.edu/~mejn/netdata/` from Prof. Mark Newman.

3 Paper list for your survey

Of course, all team members **should read Part I** of the graph mining textbook. No need to comment on it.

In addition, please choose 3 papers per person, from the list below, and comment on them:

3.1 Papers for your survey

You may survey more than 3 papers per person, but check with the instructor first.

Anomaly detection, and scoring

1. Graph anomaly detection survey by Leman Akoglu et al. [2]
2. OddBall paper (Akoglu et al) [1]. Spots nodes that have strange 'ego-nets', in weighted or unweighted settings.
3. isolation forests (Liu+, ICDM'08) [12]. Gives a 'weirdness' score to each point in a k-dimensional cloud of points.

4. random cut forests Guha+, ICML'16. [7] Similar to Isolation Forests.

Spectral methods for lock-step behavior

1. Spectral methods: EigenSpokes and the 'SpokEn' algorithm; [13] 'LockInfer' follow-up algorithm. Both spot groups of nodes that have similar behavior, which is usually suspicious.
2. Dense-block detection algorithms: Fraudar (Hooi+, KDD'16) [9] D-cube [17] and M-zoom (Kijung Shin et al) [16] Like the spectral methods, but have simpler algorithms, often have better accuracy, and give probabilistic performance guarantees.
3. CopyCatch (Beutel+, www13) [3]. Finds nodes with lock-step behavior, taking timestamps into account.
4. CatchSync (Jiang+, kdd'14) [10]. Finds groups of nodes that are (a) too similar to each other and (b) too different from everybody else.
5. ND-Sync - summary outside paywall (Giatsoglu+, PAKDD'15) [4] and also [5]. Algorithms to spot strange groups of twitter users.
6. f-Box [15]. Complements spectral methods, spotting small groups of suspicious nodes that may be missed otherwise.

Applications, explainability

1. Human-trafficking detection (Rabbany+, KDD'18) [14].
2. Spectral Lens Goebl+, ICDM'17 [6] Focuses on weighted graphs.
3. LookOut LookOut (Gupta+, PKDD'18) [8]. Gives algorithms to visually justify the outliers that, say, isolation forests, have discovered.

4 Tasks and Deliverables

Here is the detailed list of deliverables and point distribution. The maximum grade in each phase is 100, and the weights of each phase are as announced (10%, 10% , 80%)

Phase 1: 10% of project weight, max score: 100

Your write-up should be about 6-8 pages.

1. (**30 pts**) Complete a literature survey: at least *3 paper reviews per team member*, from the introductory papers above (Section 3). Paper reviews should consist of
   (a) the *problem definition* that the paper is addressing
   (b) a *summary* of the main idea of the paper (in your *own* words - cutting-and-pasting text from the paper or any other source, is **plagiarism**)
   (c) whether/why it is *useful* for your `GraphDetective` project.
   (d) list of *shortcomings*, that you think that future research could address.
2. (**35 pts**) Implement and provide all the plots that Perseus gives (degree distribution, etc), both for the in-degree, as well as the out-degree.
   Must do

- Do mark the top k=5 outliers in each plot (use isolation forests, or random-cut-forests.)
- Use heatmap-producing code (eg., `hexbin` of python), or the code from Prof. Danai Koutra - `http://www.cs.cmu.edu/~christos/SRC/heatplot.tar`, to handle over-plotting
- Use the (excellent) `networkx` library of python.
- Consider `dash - plotly` for interactive scatter plots.

No need to do

- No need to make it interactive (yet); and thus
- no need to provide ego-nets (that is, the bottom panels of Figure 1).

3. (**25 pts**) Write (at least) 3 unit tests for each of the panels of Perseus - use `pyunit`. The unit tests could be, say: a 5-node-clique graph as the input, a 10-node chain, etc. Use simple graphs for which it is easy to infer and check properties.
4. (**10 pts**) Report the suspicious nodes that you have found, on the *'MysteryDataset - unweighted'* dataset. Give the list of such nodes, grouped, if they form natural groups (eg., nodes of a suspicious near-clique should be reported together; similarly, the nodes of a suspicious chain, etc). Also give the plot that supports your decision to report them.

Phase 2: 10% of project weight, max score: 100

Your write-up should be about 10-15 pages (**including** your Phase 1 write-up)

1. (**40 pts**) Add 3 or more panels to your `GraphDetective` system, to handle weighted graphs. It is up to you to decide what these panels should have. Ideally, they should have whatever plots helped you find anomalies in the *'MysteryDataset - weighted'* dataset.
2. (**20 pts**) Start making your system interactive: When the user clicks on a 'red point' (= outlier), your system should provide a list of 1 or more nodes, that correspond to that outlier point.
3. (**40 pts**) Run your code on the *'MysteryDataset - weighted'* dataset, and report anomalies:
   - give the list of node-ids that you think are suspicious, grouped accordingly
   - justify our decision, with words and with plots.

Phase 3: 80% of project weight, max score: 100

Your write-up should be about 20-30 pages long (including all previous write-ups).

1. (**40pts**) Bug-fixing for the unweighted and weighted case; overall system implementation and correctness. It should run on several test-graphs that we will reveal after the due date. and/or on synthetic graphs of your own.
2. (**10 pts**) Stress-test your system, and report results on one graph of your choice from the datasets of Section 2 on page 3. Give the k=2 most suspicious (groups of) nodes, along with your justification (plots, and arguments).
3. (**30pts**) Make your system even more interactive: In Phase2, when the user clicks on one of the 'red' points (= outliers), the system should provide a list of 1 or more suspicious nodes. Here, when the user clicks on a node in that list, the system should give the ego-net of that node (either as a spring-model, if the ego-net is small, or as the adjacency matrix,

with careful ordering of the rows/columns - say, by pageRank, or by some other way of your own invention).

4. (**5 pts**) Provide user-manual documentation of your `GraphDetective` system, in 1 page or less: how to install it, how to run it.
5. (**15 pts**) Packaging of code and documentation: Provide a tar-file with your code, the unit-tests, and your report (latex sources). Make sure it matches the check-list in subsection 5.2.

5 Details on deliverables and software packaging

5.1 Check-lists on deliverables:

For every phase, please:

1. Hand-in a hard copy of your write-up, typed, 12pt font, neat and with pictures if applicable
2. and a tar-file with your code, including a `makefile`

More details:

1. Use the LaTeX template at:
   `http://www.cs.cmu.edu/~christos/courses/826-resources/PROJECT-SAMPLES/samplePaper.tar.gz`
   Adapt the section headers, accordingly, eg.,
   - introduction
   - ph1: paper-reviews by person1
   - ph1: paper-reviews by person-2
   - ph1: description of unit tests
   - ph2: description of algorithms
   - ph3: exceptions and anomalies
   - etc
2. Provide a plan of activities and time estimates per group member.
3. List which group member did (or will do) what.
4. Check grammar and syntax (small penalty for each typo/grammar error).
5. Keep the graded reports and attach them, every time. That is for Phase 2, attach the graded Phase 1 report; for Phase 3, attach all previous, graded, reports.

5.2 Code packaging and ease-of-use

Being a mainly implementation project, your code should be nicely packaged - follow the template at
`http://www.cs.cmu.edu/~christos/courses/826-resources/PROJECT-SAMPLES/samplePackage.tar.gz`

That is, you are expected to deposit a `tar`-file, matching the following check-list:

1. neatly packaged, ie., no extraneous files (*.o, *.pyc, *.aux, *.log)
2. able to run on the linux-andrew machines,
3. with unit-tests (`pyunit`)

4. *without* huge data files (if needed, use, e.g., `wget` in your `makefile`, to pull all such files, dynamically).
5. with a `makefile`
   - typing `make` should run a small demo;
   - `make paper.pdf` should create your report
   - `make clean` should eliminate all the derived files (*.o, *.class, *.aux, etc)
   - `make all.tar` should create a tar-file, ready for distribution
6. with brief usage instructions, in the `README` file
7. with your report in the directory `./DOC`
8. and the code should have an easy interface: e.g., from the Unix prompt
   `GraphDetective my-edge-list.csv`
   with appropriate arguments should run your system, create the all its panels, mark the outliers in each panel, etc.

5.3 Logistics - reminders

- *Academic Attribution / Plagiarism*: Whenever you use ideas, text, code, algorithms, from someone else, *please cite this person, paper, or url.* Copying without attribution constitutes **plagiarism** leading to severe penalties (failing the class, expulsion, etc).
- *Team size*: As mentioned, all projects will be done in **groups of two** (with exceptions only under special circumstances, after instructor's permission.)
- *Poster: Optional*, in contrast to the non-default projects. If you do want to do a poster, please notify the instructor.

References

[1] L. Akoglu, M. McGlohon, and C. Faloutsos. oddball: Spotting anomalies in weighted graphs. In *PAKDD (2)*, volume 6119 of *Lecture Notes in Computer Science*, pages 410–421. Springer, 2010.

[2] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data Min. Knowl. Discov.*, 29(3):626–688, 2015.

[3] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW*, pages 119–130. International World Wide Web Conferences Steering Committee / ACM, 2013.

[4] M. Giatsoglou, D. Chatzakou, N. Shah, A. Beutel, C. Faloutsos, and A. Vakali. Nd-sync: Detecting synchronized fraud activities. In *PAKDD (2)*, volume 9078 of *Lecture Notes in Computer Science*, pages 201–214. Springer, 2015.

[5] M. Giatsoglou, D. Chatzakou, N. Shah, C. Faloutsos, and A. Vakali. Retweeting activity on twitter: Signs of deception. In *PAKDD (1)*, volume 9077 of *Lecture Notes in Computer Science*, pages 122–134. Springer, 2015.

[6] S. Goebl, S. Kumar, and C. Faloutsos. Spectral lens: Explainable diagnostics, tools and discoveries in directed, weighted graphs. In *ICDM*, pages 877–882. IEEE Computer Society, 2017.

[7] S. Guha, N. Mishra, G. Roy, and O. Schrijvers. Robust random cut forest based anomaly detection on streams. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2712–2721. JMLR.org, 2016.

[8] N. Gupta, D. Eswaran, N. Shah, L. Akoglu, and C. Faloutsos. Beyond outlier detection: Lookout for pictorial explanation. In *ECML/PKDD (1)*, volume 11051 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2018.

[9] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos. FRAUDAR: bounding graph fraud in the face of camouflage. In *KDD*, pages 895–904. ACM, 2016.

[10] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang. Catchsync: catching synchronized behavior in large directed graphs. In *KDD*, pages 941–950. ACM, 2014.

[11] D. Koutra, D. Jin, Y. Ning, and C. Faloutsos. Perseus: An interactive large-scale graph mining and visualization tool. *PVLDB*, 8(12):1924–1927, 2015.

[12] F. T. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *ICDM*, pages 413–422. IEEE Computer Society, 2008.

[13] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *PAKDD (2)*, volume 6119 of *Lecture Notes in Computer Science*, pages 435–448. Springer, 2010.

[14] R. Rabbany, D. Bayani, and A. Dubrawski. Active search of connections for case building and combating human trafficking. In *KDD*, pages 2120–2129. ACM, 2018.

[15] N. Shah, A. Beutel, B. Gallagher, and C. Faloutsos. Spotting suspicious link behavior with fbox: An adversarial perspective. In *ICDM*, pages 959–964. IEEE Computer Society, 2014.

[16] K. Shin, B. Hooi, and C. Faloutsos. M-zoom: Fast dense-block detection in tensors with quality guarantees. In *ECML/PKDD (1)*, volume 9851 of *Lecture Notes in Computer Science*, pages 264–280. Springer, 2016.

[17] K. Shin, B. Hooi, J. Kim, and C. Faloutsos. D-cube: Dense-block detection in terabyte-scale tensors. In *WSDM*, pages 681–689. ACM, 2017.