A system for collaborative visualization of large network layouts using FURY and sockets.io

Personal Info

Name: Bruno Messias

University: PhD student at University of São Paulo, Brazil

github: https://github.com/devmessias

https://devmessias.github.io

Why me?

I have the following qualifications: BS, MS in theoretical physics, and computational methods. Today I'm pursuing a Ph.D. in the intersection between computer science and computational physics. Specifically, I'm studying and developing algorithms inspired in mathematical physics to characterize large graphs for social network analysis. I have also experienced working on industry and social projects related to computer science. During my master's degree, I created with my co-advisor a software which is used by thousands of scientists in Brazil.

Technical Knowledge and experience

(2016-) - Cofounder of Python Triângulo Community I'm cofounder of *Python Triângulo in Minas Gerais in Brazil*. https://pythontriangulo.github.io/

(2016-2017) - Computer literacy:I took part in a project on teaching recycling trash collectors from cooperatives basic technology and computer skills. Relevant Open Software Projects

(2018-2020) eMaTe Package: eMaTe it is a python package which provides methods capable of estimating the spectral densities and trace functions of large sparse matrices. eMaTe can run in both CPU and GPU and can estimate the spectral density and related trace functions, such as entropy and Estrada index, even in directed or undirected networks with million of nodes. https://github.com/stdogpkg/emate. **Languages and Libs:** CUDA, Python, Tensorflow(2018-2020)

(2016-2020) CapesRedir: Due to technical decisions in the Brazilian government system, many scientists across the universities in Brazil have had difficulty accessing papers. Therefore, my master co-advisor and I have developed an extension to facilitate such access, which is used by thousands of scientists in Brazil.

https://chrome.google.com/webstore/detail/redirecionamento-capes-pe/lpfcdddcpijdfghkimmcpkmidafnljbo?hl=pt-BR. **Languages and Libs::** Javascript, CSS

Project

Sub-org name: FURY

Abstract

Data structures that complex networks (graphs) can model are present almost everywhere. Therefore, it is important to have a software that can visualize and provide insights into such structures. Unfortunately, the vast majority of network visualization and manipulation softwares currently available has several limitations. For example, Gephi (Java) does not scale well when the number of nodes increases and cannot deal with 3d layout algorithms natively. Because of those limitations in the currently available network visualization softwares, we propose to develop a client/server network visualization for FURY. Our proposed system is inspired by the following work: *"Harnessing WebGL and WebSockets for a Web-Based Collaborative Graph Exploration Tool"*[1].

Technical requirements

We want that our proposed system have the following technical requirements:

- It should be able to render 1 million of nodes with an excellent performance in a low-end graphics cards (2GB VRAM)
- It should be able to render nodes positions using different layout algorithms
- It should be able to distribute changes to the network structure to all connected clients (front-end) in real time and send the rendered image.
- It should be easy to connect an external system with the server and change the stored network structure.

NOTE: I already have a sample project in which I implemented some components (Socket.io server and client). https://github.com/devmessias/stdogviz/tree/master/python. Although it is not efficient and uses webgl instead of FURY

Project timeline

First Phase

Choose, implement and integrate a set of layout algorithms to FURY.

Specifically, we want to consider algorithms with a small computational cost, ~ O(|V| log |V||). The major reason to choose those algorithms is that we want to visualize sparse networks with millions or hundred thousands of nodes.

Recently a set of these network embedding algorithms has been implemented efficiently in Python with GPU support (see [2] and [3] for more details). Therefore, part of this first phase can be performed **quickly** because it will only be necessary to integrate these methods with FURY.

libs and languages: python, FURY, cuGraph, pymde [2, 3]

June 7 , 2021 - June 11, 2021

- Pymde FURY integration.

For more details about Pymde and Minimum distortion embeddings see [2, 3]

June 14 , 2021 - June 18, 2021

- cuGraph (from RapidsAI) ForceAtlas2 integration

For more details about cuGraph-forceAtlas2 see [4, 5]

June 21 , 2021 - June 25, 2021

- Benchmarking analysis.
- First phase documentation with suggestions, issues, and bottlenecks.

Second Phase

This phase will be responsible to create the back-end server for the visualization tool

libs and languages: python, Flask, flask-socketio[6]

June 28 , 2021 - July 2, 2021

- Choose the proper way to store the network data
- Prototype the API endpoints which will allow the user and external systems to change the stored network

June 5 , 2021 - July 9, 2021

- Implement the Socket.io server which will be responsible to store and render the network.

June 19 , 2021 - July 23, 2021

- Integrate the Socket.io server with FURY layout algorithms created in the first phase of the current project.

June 19 , 2021 - July 23, 2021

- Benchmarking analysis.
- Second phase documentation with suggestions, issues, and bottlenecks.

Third Phase

Front-end/User Interaction

libs and languages: python, js   python-socketio[client], socket.io(npm)[7]

Here we call a front-end system any system which can receive the rendered image, send actions through the socket to modify the stored network data in the back-end and receive notifications.

June 26 , 2021 - July 30, 2021

- Implement the python front-end client, which will receive the rendered image and send actions to the backend.

Here, it's an example of a python client which I've implemented using sockets.io

https://github.com/devmessias/stdogviz/blob/master/python/stdogviz/main.py

August 2 , 2021 - August 7, 2021

- Implement the web frontend client, which will receive the rendered image and send actions to the backend through sockets.
- Here, it's an example of a front-end python client which I've implemented using sockets.io
- https://github.com/devmessias/stdogviz/blob/master/python/stdogviz/main.py

August 9 , 2021 - August 14, 2021

- UX analysis
- Third phase documentation with suggestions, issues, and bottlenecks.

REFERENCES

[1] Zimmer B, Kerren A. Harnessing WebGL and WebSockets for a Web-Based Collaborative Graph Exploration Tool. In: Cimiano P, Frasincar F, Houben G-J, Schwabe D, editors. Engineering the Web in the Big Data Era [Internet]. Cham: Springer International Publishing; 2015 [cited 2021 Apr 12]. p. 583–98. (Lecture Notes in Computer Science; vol. 9114). Available from: http://link.springer.com/10.1007/978-3-319-19890-3_37

[2] cvxgrp/pymde [Internet]. Stanford University Convex Optimization Group; 2021 [cited 2021 Apr 12]. Available from: https://github.com/cvxgrp/pymde

[3] Agrawal A, Ali A, Boyd S. Minimum-Distortion Embedding. arXiv:210302559 [cs, math, stat] [Internet]. 2021 Mar 20 [cited 2021 Apr 12]; Available from: http://arxiv.org/abs/2103.02559

[4]. Linsenmaier H. Large Graph Visualization with RAPIDS cuGraph [Internet]. Medium. 2020 [cited 2021 Apr 12]. Available from: https://medium.com/rapids-ai/large-graph-visualization-with-rapids-cugraph-590d07edce33.

[5] rapidsai/cugraph [Internet]. RAPIDS; 2021 [cited 2021 Apr 12]. Available from: https://github.com/rapidsai/cugraphi

[6] socket.io-client [Internet]. npm. [cited 2021 Apr 13]. Available from: https://www.npmjs.com/package/socket.io-client

[7] Welcome to Flask-SocketIO's documentation! — Flask-SocketIO documentation [Internet]. [cited 2021 Apr 12]. Available from: https://flask-socketio.readthedocs.io/en/latest/