Google Summer of Code 2023 - NetworkX

Contributor Proposal

Author:
Kavish Senthilkumar
University of Pennsylvania

April 4, 2023

Interactive Pedagogical Notebooks and `nx-guides` Contributor Guide

1. Introduction

About Me

I am a rising 4th year at the University of Pennsylvania in Philadelphia, Pennsylvania, United States studying Computer Science and Mathematics, with a concentration in Networked Systems.

Motivation

Teaching is my passion —I aim to pursue a Ph.D. in computer science (focusing on graphs) and become a professor.

I enjoy teaching because I love the challenge of translating complex ideas into understandable concepts for other people. This project is perfect for that: I can focus my time on explaining a set of algorithms as well as possible and gain experience teaching graph theory.

In addition, I myself am the target audience for this project. In my computer science classes, I love deep dives and exploratory projects that teach me more about topics. They are what made me initially interested in delving into computer science and considering graduate study. In fact, one of my professors (Dr. Kannan) would always take asides and discuss graph theory and his work, and I credit him for piquing my interest in networks. I would like to put significant effort into the notebooks because they are exactly what would have captured the interest of people like me: I want to pay it forward to the community.

With this project, I can stir the minds of students/readers and encourage intellectual exploration. I am especially excited about teaching graph theory, as it is my favorite subject. I appreciate the applications of graphs and find their properties elegant.

Relevant Experience

I am fluent in Python and have extensive experience with Jupyter Notebook. I have a portfolio of several exploratory projects in Jupyter Notebook (Link).

I am currently taking a comprehensive course in network theory. I have completed coursework in introductory and advanced graph theory. I have implemented my own suite with graph representations and algorithms in Java.

I am an experienced computer science instructor: I have been a teaching assistant and lecturer for CIS120: Programming Languages and Techniques (syllabus) for the past 3 years, and have taught algorithms and data structures as a private tutor for the last year. I have created educational YouTube videos about graph theory as well—these include an explanation of Gale-Shapley matching (Link) and a graph theory exercise walkthrough (Link).

Otherwise, my background is in academic research and open source development. I have conducted research for 6 years primarily in computational and network biology (ecological informatics with machine learning @ Brisson Lab, genetic networks @ Balin Lab and Xing

Lab). I am the development lead for the world's first open source virtual reality surgery simulator at Penn Medicine (Demo).

I have used NetworkX before to study clustering in genetic interaction networks.

Contributions

I've spent time getting familiar with the `nx-guides` codebase—I've read through all current guides and the tutorial. I've sent in PR #98 (improving consistency as mentioned by @RossBar in the community meeting on 3/21/23) and #99 (adding notes on format to a past contributor's guide). I've attended multiple NetworkX community meetings, and it's been great getting to meet you all!

I've also built out my blog for hopefully posting my contribution updates over the summer.

Statement of Commitment

My contributions to NetworkX will be my first priority. I will be working for NetworkX over the summer, during which I have no classes. The only other task I will be doing is working part-time in a research lab studying computational gene networks (10-15 hours/week). The rest of my time will be devoted to the project. I am looking for a 175 hour project, but am open to working a bit longer on the project if needed.

Contact Information

I prefer communication via email.

Email: kavishs@seas.upenn.edu
Mobile Number: +1 610-395-0526
LinkedIn

Interactive Pedagogical Notebooks and `nx-guides` Contributor Guide

Abstract

NetworkX is a Python package offering a large set of algorithms and tools for the study and manipulation of graphs. Although the graph algorithms are well-documented, users will benefit from explanations and example uses of the algorithms. Specifically, these explanations and uses will 1) provide a deep understanding of the algorithms and best practices, 2) encourage interested readers to dive further into the study of networks and graph theory, and 3) provide guidelines/templates for users to conduct their own analyses.

As such, I will be creating pedagogical Jupyter Notebooks that clearly explain and experimentally explore (with examples) 3 general algorithms implemented in NetworkX: shortest path, matching, and clique-finding. The notebooks will be published at `networkx/nx-guides` to make them widely available to users.

I understand NetworkX may be looking to have multiple contributors adding notebooks, and there may be conflicts with more than one person looking to do the same topic. To avoid this, I have provided an extra alternative topic that interests me (trees). NetworkX can replace any one of my topics with this alternate if it works better with other contributors.

In addition, new contributors currently do not have a roadmap for contributing to `nx-guides`, as mentioned by @MridulS in community meeting notes. In addition to my pedagogical notebooks, I hope to finalize a easy-to-use contributing guide for `nx-guides`.

Technical Details

I will divide this discussion into four categories: one for each proposed notebook (including the alternative notebook). For each notebook, I will focus on explaining and visualizing the steps of the algorithm (Simonak et. al show that algorithm visualizations are an important tool in helping computer science students understand complex algorithms).

In addition, I will use practical examples and aim to build intuition of the reasoning behind the algorithm. I will address the audience as if they have a basic background in graphs and notation (directed vs. undirected, nodes, vertices), experience with runtime analysis and data structures, some experience with mathematical rigor, and Python experience. I will follow a consistent format (one that I will design in the contributor guide and have already mentioned a bit in PR #99), including an import packages section, a references section, etc.

1. Shortest Path Algorithms

NetworkX provides multiple graph algorithms. I will be focusing on 3 in this pedagogical notebook: Dijkstra's, Bellman-Ford, and Floyd-Warshall. For this notebook, I will aim for, at minimum, coverage of all the methods under the shortest path heading here. I will explain the algorithms as follows:

- **Dijkstra's**: Provide and draw (with `draw_planar`) a small example weighted graph G with a practical application that can be used with Dijkstra's (no negative weight edges).

I am currently interested in a map representing a network of locations, with a flight starting from a source node looking to reach another destination. In this examples, edges are distances between locations.

Introduce the single-source shortest path problem.

Ask the reader to consider how we can calculate shortest paths from a single source, then introduce Dijkstra's with pseudocode. Provide a GIF of the vertices of G updating with Dijkstra's as a visual aid. Then, use the NetworkX implementation of Dijkstra's on G and reflect on the results and their practical implications (i.e. that a plane can take some shortest path as desired). Describe other practical applications including other navigational networks, social networks, etc. Casually derive/explain the big-O expression for runtime based on the CLRS textbook, a well-known algorithmic handbook.

- **Bellman-Ford:** Add negative cycles to G (call the new graph G'). Redefine the practical application. Here I am considering to say each edge weight represents fuel loss/gain for the plane.

  Reflect on how Dijkstra's does not work on G'. Introduce Bellman-Ford with pseudocode (following the original Bellman-Ford paper), with another GIF/series of NetworkX-drawn graphs to visually explain relaxation in the algorithm (typically, the relaxation step is the hardest part to grasp for those new to this algorithm). Describe why the limitations of Dijkstra's are no longer an issue. I will then use and explain NetworkX's implementation of Bellman-Ford on G', and discuss runtime again sourced from CLRS.

- **Floyd-Warshall:** Provide a practical reason why we may be interested in shortest paths between all pairs in G' and introduce the all-pairs shortest path problem. Here, I am interested in using the example of the optimal routing problem or finding a regular expression.

  Walk the reader through a way to solve the all-pairs shortest path problem: the Floyd-Warshall algorithm. Introduce the algorithm with pseudocode and a GIF/series of images as a visual aid. Break the explanation of the changes in this algorithm into multiple chunks: all at once, the pseudocode is hard to parse.

  Again, demonstrate and explain NetworkX's implementation of Floyd-Warshall. I will discuss runtime as per CLRS

After introducing these three algorithms, add a comparison section discussing the differences, similarities, and use cases for the shortest path algorithms.

2. Matching Algorithms

I will use a small bipartite graph B to provide some basic definitions of matchings, maximal matchings, and perfect matchings. I will aim to use B to demonstrate a practical application: currently, I'm thinking of a matching of friends or a matching of donors to recipients.

I will use pseudocode to explain NetworkX's greedy algorithm for finding maximal matchings (`maximal_matchings`) and provide a visualization. I will then apply the algorithm to B with NetworkX and visualize the output. I will briefly discuss runtime and aim for intuition.

I will continue to explain what a maximum-weight maximal matching is and use pseudocode and diagrams to explain NetworkX's implementation (`max_weight_matching`) drawn from Galil (1986). The guide will provide clear intuition for alternating and augmenting paths in Galil's method and briefly discuss runtime as discussed in the paper. I will demonstrate the NetworkX implementation as well. Then, with a modified version of B and visual aids, I will show the rationale behind using an edge modification to transform the maximum-weight maximal matching algorithm to a minimum-weight maximal matching (`min_weight_matching` in NetworkX) algorithm and demonstrate applying that algorithm in NetworkX. I will again visualize the results.

I will continue to describe practical uses for matching, specifically by discussing problems it can help solve.

3. Clique-Finding Algorithms

I will create a small graph G, use it to explain the clique problem, and explain why it is difficult to solve efficiently. I will use an example from systems biology, where the notion of a clique provides practical relevance in terms of a gene-interaction network.

Afterwards, I will explain the algorithm from Zhang et. al (2005) used to enumerate all cliques (`enumerate_all_cliques` in NetworkX) with pseudocode and visual aids. I will provide a discussion of candidate vertices, pruning, and maximal cliques in Zhang's method. I will briefly discuss runtime at a higher level, and focus most mathematical rigor towards explaining the algorithm.

I will repeat for the algorithm from Bron and Kerbosch (1973), as adapted by Tomita, Tanaka and Takahashi (2006) to find all maximal cliques (`find_cliques` in NetworkX). In addition to casually analyzing runtime, I will provide a surface-level explanation of the recursive implementation and discuss important aspects of the algorithm, espeically pruning (as compared to Zhang's algorithm).

I will ensure I run these algorithms with NetworkX and explore their outputted cliques with visualizations. I will describe the practical explanations of these cliques.

Afterwards, I will explain the notions of a maximal clique graph, a bipartite clique graph, and clique number with G. I will use all the functions to calculate these in NetworkX and reflect on the results.

4. Alternate Topic: Trees

I will create practically relevant trees and forests (I am currently interested in DOM trees for this topic) and explore the various definitions and representations of trees in the documentation.

I will then choose one specific set of examples, and explain one approach for finding optimal branchings and one approach for finding spanning arborescences with images and a mathematical rationale. I am looking to focus on Edmond's algorithm for both, using Edmonds et. al, 1967. I will focus on an understanding of the algorithms with visualizations of steps along the way. In terms of Edmond's algorithm, I will describe runtime and its relation to the blossom algorithm, and ensure I provide visual examples and explanations for blossom structure and dual graphs.

I will continue to discuss the practical application of these tree algorithms, especially as applied to the DOM tree example I start with.

Contribution Guide

I have made initial changes to a past iteration of the contribution guide for `nx_guides` in PR #99. I aim to rehaul the guide as per the needs described by @MridulS.

There are a few sections I will include: 1) forking and procedures for a commit, 2) choosing issues for first-time contributors, 3) consistent formatting for `nx_guides` , 4) motivation for `nx_guides` , 5) FAQ.

I aim to make the process of getting started with contributing to `nx_guides` frictionless and clear.

Schedule of Deliverables and Plan

Application Period: March 20th-April 4th

I will refine my proposal and set up my development environment. I will continue reading through past notebooks and their revision histories to get a grasp of my task. I will begin collaborating with the NetworkX team and identifying areas of `nx-guides` where I can help as well as current pain points.

Community Bonding Period: May 4th-May 28th

I will begin getting an understanding of the matching algorithms and begin my Jupyter notebook draft for them. I will focus on creating a consistent, interesting final product for readers. I will connect with @MridulS and @rossbar, my mentors. I will talk to other contributors at NetworkX and work on creating communication channels with mentors.

PHASE 1: May 29th-June 10th [DELIVERABLE]

I will iterate on the Jupyter notebook for matching algorithms. I will deliver my first complete notebook draft on June 10th.

PHASE 1: June 10th-June 20th [DELIVERABLE]

I will wait for feedback and update my notebook. I aim to deliver my final draft of the matching notebook by June 20th. Note that I started with the matching notebook because I feel it will take the least time of the three: doing this enables me to get used to the workflow.

PHASE 1: Jun 20th-July 14h [DELIVERABLE]

I will iterate on the Jupyter notebook for shortest path algorithms. I will deliver my first complete shortest path notebook draft on July 14th.

Midterm Evaluation

PHASE 2: Jul 14th-July 25th [DELIVERABLE]

I will receive feedback and update my shortest path algorithm notebook. I will deliver my final draft of this notebook by July 25th.

During this time, I will also finish my contribution guide first draft (it will be ready by July 25th) and submit it for review. I hope to be working on this guide incrementally throughout my schedule whenever I am ahead or waiting for feedback.

PHASE 2: July 25th-Aug 10th [DELIVERABLE]

I will iterate on the Jupyter notebook for clique finding algorithms. I will deliver my first complete clique algorithm notebook draft on August 10th.

I will also submit my final draft for my contribution guide.

PHASE 2: Aug 10th-Aug 21th [DELIVERABLE]

I will continuously get feedback and update my clique finding notebook. I will deliver my final draft of this notebook August 21st.

FINAL WEEK: Aug 21st-Aug 28th [DELIVERABLE]

I will work on prepping and delivering my final presentation. I will push all final changes to the repository and complete my project. If there is time, I am interested in a side project proposed by @mjschwenne: creating a notebook discussing the creation of a DAG representing NetworkX's dependency structure.

References

Hagberg, A., Schult, D., Swart, P. (2008). Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conference (pp. 11-15). Retrieved from https://conference.scipy.org/proceedings/SciPy2008/paper$_2$/

Hagberg, A., Schult, D., Swart, P. (2021). NetworkX documentation. Retrieved April 4, 2023

Hagberg, A., Schult, D., Swart, P. (2021). NetworkX GitHub repository. Retrieved April 4, 2023, from https://github.com/networkx/networkx/

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271.

Edmonds, J. (1965). Paths, trees, and flowers.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press.

Bellman, R. (1958). On a routing problem. Quarterly of Applied Mathematics, 16(1), 87-90.

Ford, L. R., Jr., Fulkerson, D. R. (1956). Maximal flow through a network. Canadian Journal of Mathematics, 8(3), 399-404.

Hopcroft, J. E., Karp, R. M. (1973). An n5/2 algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing, 2(4), 225-231.

Gabow, H. N. (1976). An efficient implementation of Edmonds' algorithm for maximum matching on graphs. Journal of the ACM, 23(2), 221-234. Gavril, F. (1972).

Algorithms for a maximum clique and a maximum independent set of a circular-arc graph. Networks, 2(4), 301-318.

Bar-Yehuda, R., Even, S. (1981). A local-ratio theorem for approximating the weighted vertex cover problem. Annals of Discrete Mathematics, 10, 27-46.

Barabási, A. L., Albert, R. (1999). Emergence of scaling in random networks. Science, 286(5439), 509-512.

Diestel, R. (2010). Graph theory (4th ed.). Springer.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5), 604-632.

Matula, D. W., Beck, L. L. (1983). Smoothing algorithms for shortest path problems. Networks, 13(2), 237-252.

Simončič, Jozef, et al. "Algorithm Visualizations as Epistemic Games: How Teachers and Students Perceive Them." Journal of Educational Computing Research, vol. 55, no. 5, 2017, pp. 663-692.

Bloss, Jack. "An Algorithm for Finding Maximum Matchings in Graphs." Computer Science Department, University of Colorado, Boulder, Technical Report TR-72-043, 1972.