COMP702 Design & Specification

Department of Computer Science

University of Liverpool

Open-Source Temporal Networks Library

Author:
Seán O'Callaghan
201452173

Primary Supervisor:
Viktor Zamaraev

Secondary Supervisor:
Othon Michail

3rd July 2020

Abstract

This article outlines the specification and design of an Open-Source Temporal Networks library. The document consists of several key sections; an introduction to the subject area; a detailed specification of the library and what it aims to achieve; the design process and proposed architecture; a project plan for the remaining work; a summary of the necessary requirements and skills needed to complete the project. The ultimate goal of the project is to develop a convenient library for the modelling and analysis of temporal networks to aid in the study and research of many real-world applications and systems.

1 Introduction

Temporal networks are prevalent across many systems in the modern world, be it natural, social, technological, financial or industrial. A few examples of temporal networks include public transport[4], human interaction[5], stock markets[2] and wireless sensors[3]. These examples show that a diverse amount of systems can be modelled as a temporal network, or are inherently so. The analysis of such networks can yield new and useful information that was previously not obvious from the raw data (for example, message logs). A convenient library which allows for such network modelling and analysis would prove very useful for multiple disciplines and domains.

A static network can be represented as a graph, consisting of nodes and edges. In its simplest form, a graph is a set of points interconnected with lines, where each line connects a pair of points. The graph connectivity is defined in the incidence function, which associates edges and node pairs (ch01 [1]). Mathematically, a graph G is an ordered triple of nodes, edges and the incidence function:

$$V = (a, b, c)$$

$$G = (V, E, \psi) \qquad E = (x, y, z)$$

$$\psi(x) = (ab),\ \psi(y) = (ac),\ \psi(z) = (bc)$$

A graph may be undirected, where an edge z connecting nodes b and c does not specify a direction, meaning the edge is valid for both directions bc and cb. Conversely, a directed graph (digraph) has directional edges. Like the undirected graph, the incidence function describes the relationship between edge and nodes;

$$\psi(xy) = (x, y)$$

where edge xy connects nodes x and y, with node x being the source and node y the sink (ch10 [1]). The edges of the graph can hold information (such as weights) that describe the relationship/connections between nodes. Such a graph is known as a labelled graph. It is important to note that modelling a temporal network is fundamentally different from a static network, i.e. you cannot simply generalize a concept/model based on a static network into a temporal one (s2.1 [11]).

In a temporal graph, each edge has a label which indicates some measure of time (discrete or continuous). This could be the start time(s) of when the edge is active, along with a duration of how long the edge is active. A temporal graph can be described as a graph whose connections change over time. The nodes of the graph remain static, while the edges update as time changes. A temporal graph can also be perceived as sequence of static graphs with an (unchanging) set of nodes. Other edge labels could correspond to temporal information if interpreted correctly (s2 [6]). In the discrete

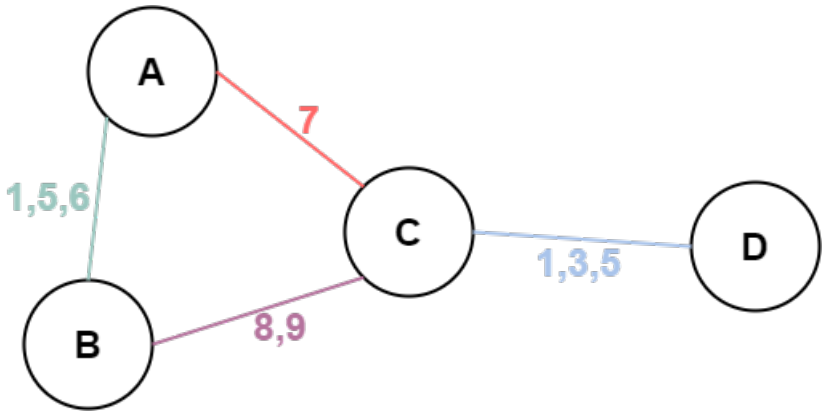

Figure 1: An static graph illustration of a temporal network.

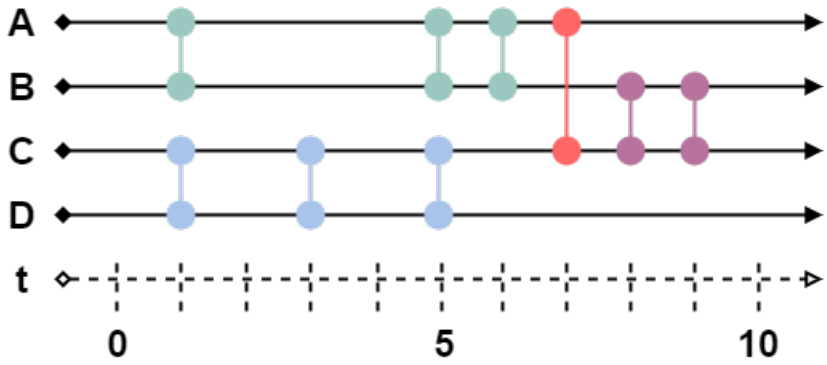

Figure 2: An timeline illustration of a temporal network.

time case, the temporal graph can be presented as a static graph $G = (V, E)$ with each edge labelled with a list of natural numbers corresponding to when that edge is active (figure 1). This time-edge (or contact/link) triple $e = (i, j, t)$ of source, sink and active time are the fundamental building blocks of temporal graphs.

The above figures illustrate a temporal network in two representations. The first figure is a static graph, with each edge labelled with the active/available times (time-edges/links). The second figure shows the timeline of the network, with the times-edges become available represented as connections between the node rows. The timeline representation better illustrates the temporal connectivity of the graph, while the static representation shows the structure of the network. Note that in the static graph, while node D is 'connected' to node A,B through C, it is in fact not possible to traverse from A/B to D, as the time-edges from C to D are, in a sense, too early. This indicates the lack of temporal information/intuition a static representation has, and further indicates that static and temporal networks are indeed fundamentally distinct. Once a temporal graph is built, there are many measurements and algorithms that can be applied to the network in order to produce temporal metrics. Network latency, reachability and centrality are all distinctive properties of temporal networks, and can have a large impact on the performance of the network being modelled.

The research and interest in the domain of temporal networks is increasing as technology advances, and there is growing evidence [12] that the modelling and analysis of such networks can produce useful results and information. Therefore, a convenient and purposeful library to enable this effort would be advantageous.

2 Specification & Aim

The overall aim of the project is to build an open-source library that enables temporal network modelling and analysis in an intuitive, user-friendly and powerful manner. The library will also allow for multiple contributors to develop the project further. Bearing this in mind, it is important for the code to be designed in a modular way. This can be achieved by using class inheritance and by defining the classes well to avoid a 'super-class' situation. The specification of the library can be split into some key areas; core functionality, extendable functionality, selected language/platform and open-source setup.

2.1 Core Functionality

The core functionality of the library is divided into several items.

- Input handling:
  Raw temporal data can be acquired and stored in a multitude of manners and formats. This is unsurprising considering the wide scope of temporal networks. It is therefore important that the library has the ability to handle multiple types of input while also allowing for new input handling methods to be added easily. There is also the possibility of having input data with no clear set of nodes or edges specified, that needs a conversion process to first decide the nodes & edges, along with the time labels and other information. An example of this could be messaging logs from a social network, or a transport network timetable.

CSV	GraphML	GEXF
GDF	GML	XML

Table 1: Possible input formats [7].

- Graph creation/generation:
  The graph can be created in two ways; either from the post-processed input data, or a generated graph from a model. A generated graph could simply be a randomly generated set of time-edges labelled from within some probability distribution, or from a set of linear functions (s7,s8 [6]).

- Graph functions:
  There are multiple functions that can be developed once the graph has been created. These can range from selecting a node or edge, retrieving information from the graph/nodes/edges, adding/removing edges/nodes and so on.

- Graph visualization:
  The created graph should be discernible in a intuitive and clear manner.

- Graph analysis:
  The user should be able to collect measurements on the graph, such as centrality, reachability and latency. The user should also be able to run and test algorithms on the graph, such as computing foremost time [8].

- Output handling:
  Any metrics generated from the graph should be presentable in a report or file (.csv, for example).

2.2 Extendable Functionality

While the library will offer a lot of functionality, as described in the previous section, it is important for the design architecture to allow for easy extension from the core functionality. This is especially true in the case of an open-source project, where there is the possibility of contribution from the community. For example, if an individual wanted to add a new algorithm to calculate the shortest path, then this should be easily integrated in the current core library.

2.3 Language & Platform

The selected language for this library is Python. Python is one of the most popular languages amongst developers and has many useful libraries on offer, such as numpy, matplotlib, scikit, pandas, tensorflow and more. All the source code and documentation will be hosted on GitHub, including this document, user-guides and readmes.
An important note is that this library is not aiming to be extremely efficient in order to handle large temporal graphs or data. A more appropriate choice of language for this would be C++. This library is essentially trying to emulate what NetworkX [9] has achieved for static graphs, in terms of convenience, functionality and open-source, communal deployment.

2.4 Open-source

The idea of making a project open-source is powerful since it allows people to freely contribute to the project without the usual barriers to collaboration. If there is enough interest and motivation in the project domain, the library can develop rapidly. It also allows the users to freely modify and customize the library to suit their own needs and the needs of others. Any customization that could benefit the user base can be requested to be added as a permanent feature in the next release, improving the overall quality of the library. There are a number of important steps involved in launching a project as open-source [10].

3 Design Overview

Outline in detail how the project will be designed and how you came to this decision. Show diagrams of object-oriented code design. Explain how and why design decisions were made.

4 Project Plan

Outline the timeline of the project. Include Gantt chart and reference to Gantt planning.

5 Requirements & Skills

Outline what will be required to complete the project and any skills necessary. Outline new skills expected to be acquired.

A Appendix

List of Figures

List of Tables

References

[1] J. A. Bondy, Graph Theory With Applications. GBR: Elsevier Science Ltd., 1976.

[2] L. Zhao, G.-J. Wang, M. Wang, W. Bao, W. Li, and H. E. Stanley, 'Stock market as temporal network', Physica A: Statistical Mechanics and its Applications, vol. 506, pp. 1104–1112, Sep. 2018, doi: 10.1016/j.physa.2018.05.039.

[3] K. Römer, Temporal Message Ordering in Wireless Sensor Networks. 2002.

[4] R. Gallotti and M. Barthelemy, 'The multilayer temporal network of public transport in Great Britain', Sci Data, vol. 2, no. 1, p. 140056, Dec. 2015, doi: 10.1038/sdata.2014.56.

[5] S. Lee, L. E. C. Rocha, F. Liljeros, and P. Holme, 'Exploiting Temporal Network Structures of Human Interaction to Effectively Immunize Populations', PLoS ONE, vol. 7, no. 5, p. e36439, May 2012, doi: 10.1371/journal.pone.0036439.

[6] O. Michail, 'An Introduction to Temporal Graphs: An Algorithmic Perspective', arXiv:1503.00278 [cs], Mar. 2015, Accessed: Jun. 29, 2020. [Online]. Available: http://arxiv.org/abs/1503.00278.

[7] 'Supported Graph Formats'. https://gephi.org/users/supported-graph-formats/ (accessed Jun. 30, 2020).

[8] H. Wu, J. Cheng, Y. Ke, S. Huang, Y. Huang, and H. Wu, 'Efficient Algorithms for Temporal Path Computation', IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 11, pp. 2927–2942, Nov. 2016, doi: 10.1109/TKDE.2016.2594065.

[9] 'NetworkX — NetworkX documentation'. https://networkx.github.io/ (accessed Jun. 30, 2020).

[10] 'Starting an Open Source Project', Open Source Guides. https://opensource.guide/starting-a-project/ (accessed Jun. 30, 2020).

[11] P. Holme and J. Saramäki, Eds., Temporal Network Theory. Springer International Publishing, 2019.

[12] J. Tang, M. Musolesi, C. Mascolo, and V. Latora, 'Temporal distance metrics for social network analysis', in Proceedings of the 2nd ACM workshop on Online social networks, Barcelona, Spain, Aug. 2009, pp. 31–36, doi: 10.1145/1592665.1592674.

[13]

[14]