Proceedings of the 11th International Conference on Applied Informatics
Eger, Hungary, January 29–31, 2020, published at http://ceur-ws.org

Implementing a New Interface for Directed Graph Analysis by Existing and New Algorithms

Miklós Becsei, Máté Csongor Széll, Gergely Kocsis

University of Debrecen, Hungary, Faculty of Informatics,
Department of IT Systems and Networks
kocsis.gergely@inf.unideb.hu

Abstract

We have developed a multiplatform application that provides an easy-to-use alternative for structural analysis of directed graphs. The work consisted of two major parts. On the one hand, a Java package has been created which, in addition to the implementation of some basic and some moderately complex algorithms, allows for the modular addition of new algorithms. On the other hand, we have created a web interface that can analyze the uploaded graphs online and display the results as a web application. The source of the Java package can be used without a web interface, so it can be easily adapted to a third-party application, or even create a new interface (e.g. in JavaFX). In addition to statistics on simple nodes and edges related to the graph, the package is able to find the giant component of the graph using the Tarján algorithm and use the result to map the so-called "tendrils" in the graph [1]. It can also specify the number of triangles of different directions in the graph [2]. The web interface gives the ability to register the user so that it can retain previously uploaded graphs and results in a consistent manner. After logging in, one can upload a new graph, delete an older one, and run the Java package algorithms on the uploaded graphs. Because this interface is independent of the underlying Java package, it can run all algorithms implemented in it and display the result, even after adding new algorithms. In the current stage of the work we are adding new algorithms by integrating well-known graph analysis software into the package under our common interface. This paper focuses on the web-interface while the details of the core package are to be published later since it may need some structural reorganization. The results

of the work are continuously made available at `http://dina.inf.unideb.hu`.

Keywords: directed graph, network topology, web interface

MSC: 68W40, 68N99, 68U01

1. Motivation

One of the most popular interdisciplinary topics of recent years is network research, which seeks to provide solutions for analyzing and evaluating complex processes across statistical disciplines from statistical physics through biology to sociology. The basis of these researches is that, taking into account a real-world process, we try to describe it as a network using a graph. Finally, we model the phenomenon based on network properties. The networks used for modeling can be directional and non-directional, edges can have weight, and vertices can be parameterized in many different ways.

There are several methods for analyzing networks. From many points of view the simplest is the development of a dedicated program. This may need some already existing programming knowledge, but does not require to learn new languages. Because of this latter property still many scientists refuse to use modern network analysis tools. Instead they implement their own algorithms in their well known languages (C, Fortran, Java) from the basis [3, 4]. Another solution is to use some freely available graph analysis software. There are many such software or programming packages available such as Gephi [5], Wolfram Alpha [6], Graphviz [7], JGraphT [8], GraphStream [9] or NetworkX [10]. Unfortunately, in many cases they do not have implementations for specific algorithms out of the box. One needs to learn how to implement or at least parameterize algorithms in these. The aim of our work is to develop a software that, beside providing some basic algorithm implementations by itself, makes it easy to run our own implementations of algorithms written in common languages or implemented by well-known packages or tools. We also would like to provide the possibility for users to add their own graph analysis algorithms under this common interface.

2. The application's structure

As a first step a small core Java package has been developed by implementing a bunch of algorithms which can be used to analyze various, unique properties of a directed graphs, which are not in the main focus of the research of directed networks in order to have an application on the top of which we implement our interface. The main goal was to make this software more easy to be used by creating a web application which through the features of it can be reached. After examining the existing application and the implementation of the graph analyzer algorithms we started to investigate the tools and technologies which are appropriate to create the application efficiently.

The application is based on client-server architecture. The server is a Spring Boot [12] application leveraging the advantages of Spring Framework [11]. It provides REST endpoints for the client application [13]. The client-server connection is secured via OAuth 2.0 protocol [14] with JSON Web Tokens managed by Spring Security [15]. For data persistence it uses MySQL relational database management system through interfaces exposed by Spring Data JPA [16]. We implemented the actual user interface on Angular platform [17] using the components of PrimeNG UI kit [18]. The custom components are written in TypeScript [21] and HTML, the look and feel defined by SCSS stylesheets. For state management we used NgRx [19], which is designed specifically for Angular based on the widely used Redux pattern [20]. The full technology stack is illustrated on Figure 1.

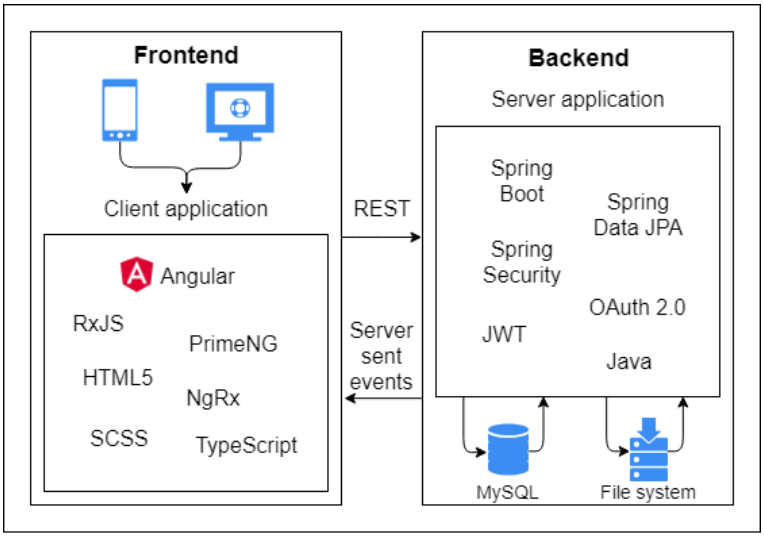

Figure 1: The technology stack used while implementing the web service interface for the core Java application.

In order to use our web-service users need to have an account. The registration process is quite simple, consists of providing a username and password secured by a CAPTCHA test. After completing the procedure, the user can log in to the application. As the server's REST endpoints are secured – except the registration and the login – the server will decline all the unauthorized requests.

3. The application's features

After the login the user can upload CSV files containing graphs to the server. At the moment only two specific formats are accepted. *i.)* The first line of a CSV file contains the number of nodes and the upcoming lines describe the links as pairs of node IDs separated by a whitespace characters, commas or semicolons. Numbering

of IDs start from 0, so if we have for example 7 nodes the ID of the first one is 0 while the ID of the last one is 6. *ii.)* The file contains only pairs of nodes in each line. In this case those nodes that have no links cannot be represented. The upload procedure is an HTTP POST request, the file is transferred in the request body. The uploaded graphs are bound to the user by a database table. During the upload process a new row is inserted into this table saving the date of the upload, the name of the file and the identifier of the uploading user. The file itself is stored in the file system. The user's uploaded graphs can be listed on the UI allowing the user to do different actions on them.

One of the main goals while implementing the interface was to implement it without the need of changing the already existing Java core. Keeping this aim in mind helped to build the interface in a modular way making it absolutely independent from the algorithms while on the other hand keeping it easy to add new features. To accomplish this, we created adapter classes to convert the input data to the proper format for the algorithms. POJOs are also made for each of them to transfer the results of the analysis.

After these preparations we started to invoke the analyzer methods with different sized graphs to measure the run times. It turned out that the processes can be really time consuming in case of big graphs with many nodes and edges. The solution for this became to run the methods in asynchronous manner and persist the results to the file system in JSON format. The advantage of this strategy is that the stored result can be pushed directly to the client without any conversions or calculations when it asks for it. Figure 2 shows the UI in a state where 3 methods have finished while the fourth one it still in progress. It is important to note that more than one method can run at the same time and also that one method can be run only once. We put information circles nest to the names of the algorithms. Pressing these show detailed description of them.

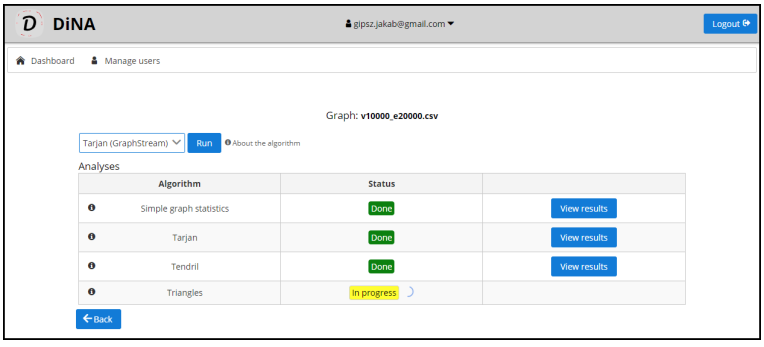

Figure 2: Three analysis methods have finished on the selected graph while one is still in progress. The information circles next to the names of the algorithms provide detailed description about them to the user.

The user can start the analysis of a graph by selecting it from the list on the

UI. At first the server will check if the result is existing for the specified graph and algorithm, if it does, it returns the result immediately, if does not, the process starts on another thread and the client will get an empty response. In that way the caller thread is not blocked so there is no infinite loading screen presented to the user. One analysis can be started one time on a graph to avoid the overloading of the server with unnecessary tasks. The results will be stored when the analysis completes, and the client is notified about this fact. When it happens, the client will send a request to the server to obtain the result and presents it to the user. If a graph contains a huge number of nodes and edges the result of the analysis cannot presented well visually for the user, so it can be downloaded from the application in text format allowing additional processing and investigation. This file is created by a conversion of the stored JSON. The uploaded graphs and the results can be deleted from the application, after confirmation all data will be physically deleted from the server if the user makes that decision. At the current state of the web-application four plus two different algorithms can be run on directed graphs: i.) basic properties are expressed such as number of nodes, links, bidirectional links (links that have directions for both directions), unidirectional links (links having only one direction), loops (links that for which the start and the end are the same), multilinks (more than one links having similar starting and end nodes), in-degree (number of incoming links) and out-degree (number of outgoing links) (see Figure 3 (*top-left*)), ii.) strongly connected components including the giant component using the Tarján algorithm [22, 23] (see Figure 3 (*top-right*)), iii.) tendrils and tubes for all layers [1] (see Figure 3 (*bottom-left*)), iv.) number of different triangles in the graph [2] (see Figure 3 (*top-right*)). The detailed description of these algorithms is out of the scope of this paper since currently we focus on the interface itself. However in the software we added a description next to each algorithm that describes all the needed details. The two extra algorithms mentioned above are two algorithms from third-party software packages that have been successfully merged to the system to test how easy it is to add already existing solutions to the package.

By successfully integrating the core Java package we have developed this interface in a way that lets us adding more (even pre-written) other algorithms from other well-known graph analysis software. We have successfully added algorithms from JGrapht and GraphhStream. Although using the built in Java interfaces of the core package adding Java implementations of algorithms is possible, we are also working on a part of the application that lets users to add their own implementations in several different languages. A powerful property of our interface is that if the user precisely implement the provided Java interfaces of the core package, that visualization types are automatically generated. Right now these may be Map, Table, Graph and File. See examples of these on Figure 3. Since this paper is about the web-interface the detailed description of this process is out of the scope. However for users who would like to try it, after requesting the code from the authors the JavaDoc of the classes provide enough help.

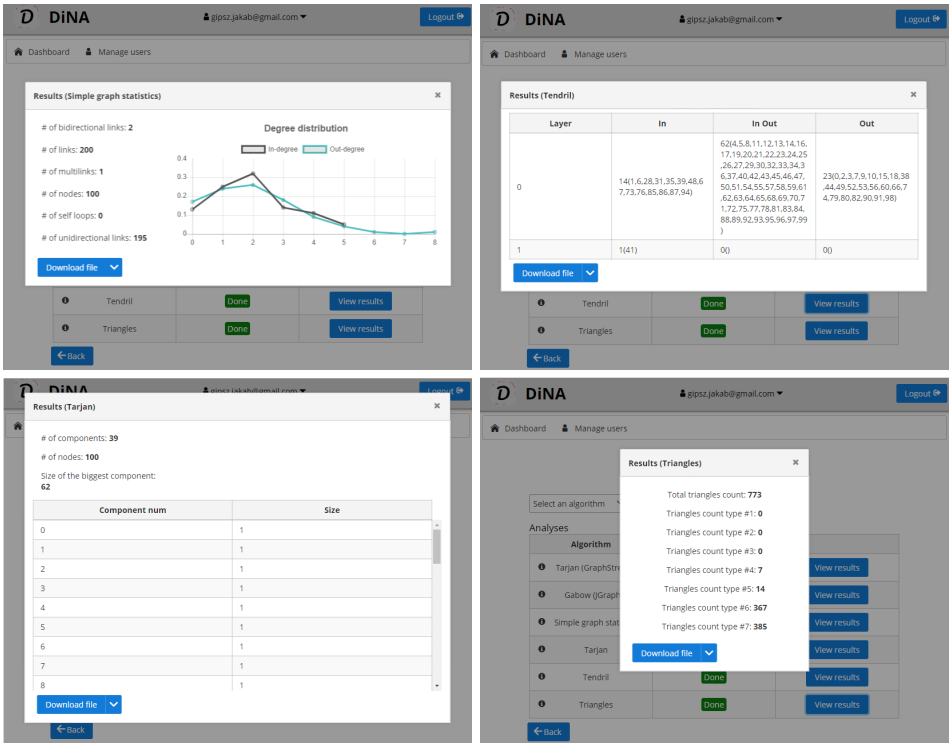

Figure 3: Snapshots of the results of the algorithms run on a sample graph. The visualization is automatically generated based on the output of the implementation of the algorithm in the core package. *(upper left)* Example of map and linechart results through simple graph statistics. *(upper right, lower left)* Examples of table result through Tarjan and Tendril algorithms. *(lower right)* Example of map results through triangle counter algorithm. In each case the results can be downloaded also in file format specified by the algorithm itself. For more detailed investigation please visit `http://dina.inf.unideb.hu`.

4. Discussion

In this work, we have developed a multiplatform application that provides an easy-to-use alternative for structural analysis of directed graphs. Based on an already existing Java core package for graph analysis we have created a web interface that can analyze the uploaded graphs online and display the results as a web application. In addition to statistics on simple nodes and edges related to the graph, the package is able to find the giant component of the graph using the Tarján algorithm and use the result to map the so-called "tendrils" in the graph [1]. It can also specify the number of triangles of different directions in the graph [2]. The web interface gives

the ability to register the user so that it can retain previously uploaded graphs and results in a consistent manner. After logging in, one can upload a new graph to the interface, delete an older one, and run the Java package algorithms on the uploaded graphs. Because this interface is independent of the underlying Java package, it can run all algorithms implemented in it and display the result, even after adding new algorithms. In the current stage of the work we are adding new algorithm implementations by integrating well-known graph analysis software into the package under our common interface. We also provide the possibility for users to add their own graph analysis algorithms under this common interface. Right now there are two ways to do this. the user can either send the implementation of the algorithm to the authors to add it to the list. Or what is more comfortable, by ask the full code of the software is sent so the new algorithm can be added on the locally deployed version. In the future we would like to make this more easy by providing scripts for the re-deployment.

Acknowledgements. Miklós Becsei and Máté Csongor Széll were supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

Gergely Kocsis is supported by the EFOP-3.6.1-16-2016-00022 project. The project is co-financed by the European Union and the European Social Fund.

References

[1] Timár, G., Goltsev, A. V., Dorogovtsev, S. N., Mendes, J. F. F., *Mapping the Structure of Directed Networks: Beyond the Bow-Tie Diagram*, Physical Review Letters 118, 078301 (2017).

[2] Roughgarden, T., *Reading in Algorithms Counting Triangles*, Stanford University (2014).

[3] Teng, S.-H., *Scalable Algorithms for Data and Network Analysis*, now Publishers Inc, Boston–Delft (2016).

[4] Varga, I., *Comparison of network topologies by simulation of advertising*, In: Gusikhin, O., Méndez Muñoz, V., Firouzi, F., Mønster, D., Chang, C. (eds.) Proceedings of the 2nd International Conference on Complexity, Future Information Systems and Risk (COMPLEXIS 2017), pp. 17–22. SciTePress (2017).

[5] Bastian, M., Heymann, S., Jacomy, M., *Gephi: an open source software for exploring and manipulating networks*, International AAAI Conference on Weblogs and Social Media (2009).

[6] Official site of WolframAplha: `https://www.wolframalpha.com/` (last visited: 01.15.2020).

[7] Official site of graphviz: `https://www.graphviz.org/` (last visited: 01.15.2020).

[8] Official site of JGraphT: `https://jgrapht.org/` (last visited: 01.15.2020).

[9] Official site of GraphStream: `http://graphstream-project.org/` (last visited: 01.15.2020).

[10] Official site of NetworkX : https://networkx.github.io/ (last visited: 01.15.2020).

[11] Official site of Spring Framework: https://spring.io/projects/spring-framework (last visited: 01.15.2020).

[12] Wikipedia page of Spring Framework, https://en.wikipedia.org/wiki/Spring_Framework#Spring_Boot (last visited: 01.15.2020).

[13] The official REST tutorial, https://restfulapi.net/ (last visited: 01.15.2020).

[14] Sathya Bandara, *An Introduction to OAuth 2.0* (2017), https://medium.com/@technospace/an-introduction-to-oauth-2-0-4c71b5fb19ff (last visited: 01.15.2020).

[15] Jorge Veliz, *5 Easy Steps to Understanding JSON Web Tokens* (JWT) (2018), https://thejlmedia.com/5-easy-steps-to-understanding-json-web-tokens-jwt/ (last visited: 01.15.2020).

[16] Spring Data JPA, https://spring.io/projects/spring-data-jpa (last visited: 01.15.2020).

[17] Official Angular documentation: https://angular.io/docs (last visited: 01.15.2020).

[18] Official site of PrimeNG: https://www.primefaces.org/primeng (last visited: 01.15.2020).

[19] Official NgRx documentation: https://ngrx.io/docs (last visited: 01.15.2020).

[20] Official Angular documentation / RxJS library: https://angular.io/guide/rx-library (last visited: 01.15.2020).

[21] Official Typescript documentation: https://www.tutorialspoint.com/typescript/typescript_overview.htm (last visited: 01.15.2020).

[22] Nuutila, Esko, *On Finding the Strongly Connected Components in a Directed Graph*, Information Processing Letters. 49 (1): 9–14 (1994).

[23] Micha Sharir, A strong-connectivity algorithm and its applications to data flow analysis, Computers and Mathematics with Applications 7(1): 67–72, (1981).