SOFTWARE TOOL ARTICLE

Automating the PathLinker app for Cytoscape [version 1; referees: 2 approved]

Li Jun Huang[1], Jeffrey N. Law [2], T. M. Murali[1,3]

[1]Department of Computer Science, Virginia Tech, Blacksburg, VA, 24061, USA
[2]Genetics, Bioinformatics, and Computational Biology Ph.D. program, Virginia Tech, Blacksburg, VA, 24061, USA
[3] ICTAS Center for Systems Biology of Engineered Tissues, Virginia Tech, Blacksburg, VA, 24061, USA

v1 **First published:** 12 Jun 2018, **7**:727 (doi: 10.12688/f1000research.14616.1)
Latest published: 12 Jun 2018, **7**:727 (doi: 10.12688/f1000research.14616.1)

Abstract

PathLinker is a graph-theoretic algorithm originally developed to reconstruct the interactions in a signaling pathway of interest. It efficiently computes multiple short paths within a background protein interaction network from the receptors to transcription factors (TFs) in a pathway. Since December 2015, PathLinker has been available as an app for Cytoscape. This paper describes how we automated the app to use the CyRest infrastructure and how users can incorporate PathLinker into their software pipelines.

Keywords

Network Biology, Shortest Paths, Pathway Reconstruction, CyREST API

This article is included in the Cytoscape Apps gateway.

Open Peer Review

Referee Status: ✓ ✓

	Invited Referees 1	2
version 1 published 12 Jun 2018	✓ report	✓ report

1 **Alexander Pico**, Gladstone Institutes, USA

2 **Stefan Wuchty**, University of Miami, USA

Discuss this article

Comments (0)

Corresponding author: T. M. Murali (murali@cs.vt.edu)

Author roles: Huang LJ: Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Law JN**: Supervision, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Murali TM**: Funding Acquisition, Project Administration, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: National Science Foundation grant [CCF-1617678] supported this research. The research is also partially based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the Army Research Office (ARO) under cooperative Agreement Number [W911NF-17-2-0105]. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, ARO, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Huang LJ, Law JN and Murali TM. **Automating the PathLinker app for Cytoscape [version 1; referees: 2 approved]** *F1000Research* 2018, **7**:727 (doi: 10.12688/f1000research.14616.1)

First published: 12 Jun 2018, **7**:727 (doi: 10.12688/f1000research.14616.1)

Introduction

PathLinker is an algorithm that automates the reconstruction of any human signaling pathway by connecting the receptors and transcription factors (TFs) in that pathway through a physical and regulatory interaction network[1]. In a comprehensive quantitative evaluation on NetPath pathways, PathLinker achieved higher reconstruction accuracy than several other state-of-the-art algorithms. In addition, PathLinker's novel prediction that the cystic fibrosis transmembrane conductance regulator (CFTR), an ion-channel receptor, is involved in the Wnt pathway was experimentally validated[1]. In general, PathLinker can be used to connect any set of sources to any set of targets in a given network. The PathLinker app for Cytoscape is an implementation of this algorithm. The PathLinker app[2] was first released in the middle of December 2015. While it is possible to use the PathLinker app in conjunction with other Cytoscape apps, it can be cumbersome to create such workflows using the Cytoscape user interface. It is also challenging to reproduce the results of these workflows.

Here we present an CyREST-based API that allows users to incorporate PathLinker algorithms into their own software pipelines. The PathLinker application programming interface (API) will facilitate automated analysis of complex networks in reproducible workflows, including in conjunction with other CyREST enabled Cytoscape apps[3].

Methods

Implementation

The PathLinker API allows external software (written in languages such as Python and R) and tools (e.g., Jupyter Notebook) to access PathLinker functions via the REST protocol. The API follows the OSGi design pattern. The PathLinker API exposes two functions via JAX-RS annotations that allows them to be discovered by CyREST[3] and be made available to external callers via REST. We have documented these functions using Swagger annotations to meet the Cytoscape Automation documentation standards. The Swagger annotation allows the PathLinker API functions to be accessed via the Swagger user interface along with other API functions exposed by CyREST.

We substantially updated and refactored the PathLinker codebase to follow the principles of the OSGi modular design and to remove redundant code. Specifically, we refactored the code for generating the *k* shortest paths, network visualization, and functions related to the user interface (e.g., generating information in the "Result Panel") into distinct Task classes that can be managed by the Task Manager in the Cytoscape API. The current design enables us to use the same codebase for running PathLinker through the Cytoscape user-interface as well as through the REST API. This modular design will facilitate easy expansion of the app in the future, e.g., by implementing additional subnetwork-finding algorithms.

Operation

The PathLinker API is accessible directly through the Swagger user interface within Cytoscape or by using any REST-enabled client. Note that users must have installed v1.4 of the PathLinker app and Cytoscape v3.6.0 or higher. Moreover, an instance of Cytoscape must be running on the user's computer. In the "Use Cases" section, we describe a sample workflow using py2cytoscape and provide an example Jupyter Notebook.

As shown in Figure 2, the PathLinker API provides two POST functions, "/pathlinker/v1/currentView/run" and "/pathlinker/v1/networkSUID/run", which run PathLinker on the currently selected network and on the given networkSUID respectively. Both functions send user-selected source nodes and target nodes and a set of parameters to the PathLinker Cytoscape app.

The app computes the *k* shortest simple (loopless) paths that connect any source to any target in the network specified in the POST function, generates a subnetwork that contains these paths and a view of this subnetwork, creates a table in the Result Panel that contains these paths (See Figure 3b), and adds a "path rank" column to the Edge Table that contains the rank of the first path in which each edge appears. The POST functions return the computed paths, the SUIDs of the subnetwork and subnetwork view, and the name of the "path rank" column created by the app.

We summarize the parameters of the API functions and their outputs below.

API parameters

The API functions have the same set of parameters as the user can set in the user interface of the PathLinker app (see Figure 1a and Figure 1b). The user should provide these parameters in JSON format, whether they use the Swagger user interface or invoke the PathLinker API in code or via external tools. Table 1 contains an overview of the parameters, their types and a brief description. For a detailed description of the parameters, please see the Swagger documentation or documentation of the PathLinker app.

API Output

The PathLinker API functions returns a response in JSON format with the following fields:

subnetworkSUID: The SUID of the subnetwork created in Cytoscape.

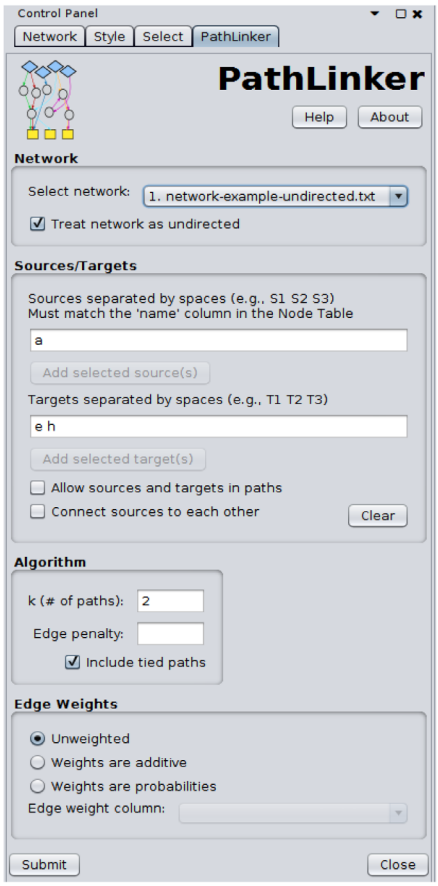
subnetworkViewSUID: The SUID of the subnetwork view created in Cytoscape.

pathRankColumnName: The name of the edge column created in the subnetwork's edge table containing the rank of the first path in which each edge appears. The user can programmatically access this table using the CyREST API.

paths: The list of paths generated by the algorithm sorted by the path rank. This list contains the same information as the result table in the "Result Panel" created by the PathLinker app. Each entry in the list contains the following fields that describe a path:

rank: The rank of the path. For example, "100" for the 100th shortest path.

score: The total weight of the edges in the path.



(a) Parameters for running PathLinker in user interface of the Cytoscape app.

(b) API parameters for PathLinker in JSON format in the Swagger user interface.

Figure 1. Comparison of PathLinker parameters in the app and in the API.

nodeList: The list of nodes in the path. Each element in this list appears in the "name" column in the node table of the network provided as input to PathLinker. If multiple paths have the same score, then we order them lexicographically by their node lists.

If the user sets skipSubnetworkGeneration to true in the input to the API functions, then the PathLinker app does not generate the subnetwork and its view. Therefore, the API output will not contain the first three attributes, i.e., subnetworkSUID, subnetworkViewSUID, and pathRankColumnName.

The functions in the PathLinker API implement careful checks of the input. The function return the following error codes to provide meaningful feedback to the user:

400: Invalid user input. The JSON response lists all faulty input along with a reason why the input is correct, allowing the user to correct all the errors at the same time.

404: Current network or network with the given SUID does not exist.

422: No paths found. This error message indicates that PathLinker could not find any path connecting the source(s) to the target(s) in the given network. This error can occur if every source is disconnected from every target.

Use cases

We provide two Jupyter Notebooks that illustrate the PathLinker API. The first notebook implements a use case on a simple network with 11 nodes and 13 edges, shown in Figure 3a. The notebook shows how to use Python and the py2cytoscape library to 1) load this network into Cytoscape, 2) call the PathLinker API with a set of parameters (Figure 1), 3) view the computed paths and subnetwork, and 4) save the paths and/or subnetwork image to a file. The API function call in the notebook asks PathLinker to find the two shortest paths connecting "a" to "e" or to "h" while treating the network as undirected. It also asks PathLinker to return more than two paths if their scores are tied with the second path.

Even though we set $k = 2$, PathLinker returned three paths; Figure 3b shows the subnetwork containing these paths. PathLinker did so because we used the "Include tied paths" option, and the second and third path both had a score of three (they contain three edges each). In this example, we used the "Treat network as undirected" option because even though the edges in the network were intended to be undirected, py2cytoscape

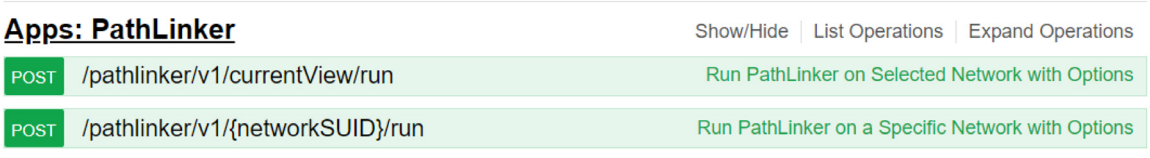
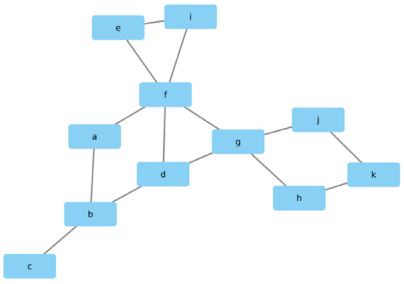
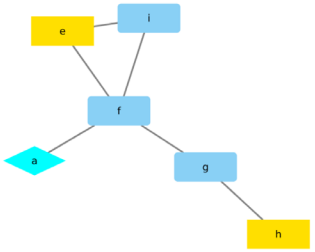
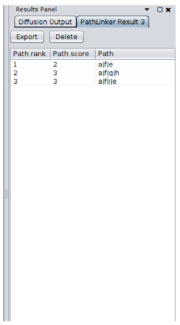


Figure 2. PathLinker application programming interface (API) functions shown in the Swagger user interface.



(a) The simple network used as input in the first use case.





(b) The three shortest paths from "a" (source) to "e" or "h" (targets) found by PATHLINKER on the example network. The subnetwork containing the nodes and edges in the three computed paths is on the left, with the Result Panel listing the paths to the right.

(c) The response from the PATHLINKER API in the Swagger user interface.

Figure 3. Illustration of the first use case.

Table 1. PATHLINKER application programming interface (API) parameters. The last column records whether an equivalent option is present in the user interface of the PATHLINKER app. T/F - True/False.

Parameter	Type	Description	Present in UI
sources	String	Node names separated by spaces	✓
targets	String	Node names separated by spaces	✓
k	Integer	Number of paths to compute	✓
edgeWeightType	String	UNWEIGHTED, ADDITIVE, or PROBABILITIES	✓
edgePenalty	Integer	Penalize low-weight paths with many edges	✓
edgeWeightColumnName	String	Name of the Edge Table column that contains the edge weights	✓
allowSourcesTargetsInPaths	T/F	Allow sources and targets to be intermediate nodes in paths	✓
includeTiedPaths	T/F	Include more than *k* paths if their score equals the *k*th path's score	✓
treatNetworkAsUndirected	T/F	Ignore directionality when computing paths	✓
skipSubnetworkGeneration	T/F	Return only the *k* shortest paths	

treats networks imported from the Python NetworkX package as directed.

The second notebook implements a more complex example that we presented in the paper describing the PathLinker app[2]. Here, we used PathLinker to compute and analyze a network of interactions connecting proteins that are perturbed by the drug Lovastatin. The PathLinker API functions now allow us to automate this use case enabling fast reproduction of the results, as shown in the Jupyter Notebook.

Summary

The subnetwork returned by the PathLinker API is another network in the current Cytoscape session. Hence, this network is amenable for analysis by other CyREST APIs and/or Cytoscape apps. Some examples include calling the Diffusion app's API on the subnetwork, or modifying the subnetwork using standard CyREST API calls such as adding visual styles to the subnetwork or calling different layout algorithms.

In the near future, we plan to provide users more freedom in determining which column in the Node Table should act as the input for source and target fields. We will also implement additional sub-network generation algorithms. We will implement these new features in parallel in the PathLinker app and in the REST API.

We hope that these additions will increase making PathLinker a method of choice in the network biology community.

Data availability

All data underlying the results are available as part of the article and no additional source data are required.

Software availability

Software available from: http://apps.cytoscape.org/apps/pathlinker

The Cytoscape app source code and sample Jupyter Notebooks are available at https://github.com/Murali-group/PathLinker-Cytoscape

Archived source code as at time of publication: http://dx.doi.org/10.5281/zenodo.1252308[4]

License: GNU General Public License version 3

Competing interests

No competing interests were disclosed.

Grant information

National Science Foundation grant [CCF-1617678] supported this research. The research is also partially based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the Army Research Office (ARO) under cooperative Agreement Number [W911NF-17-2-0105]. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, ARO, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

1. Ritz A, Poirel CL, Tegge AN, *et al.*: **Pathways on demand: automated reconstruction of human signaling networks.** *NPJ Syst Biol Appl.* 2016; **2**: 16002.
PubMed Abstract | Publisher Full Text | Free Full Text

2. Gil DP, Law JN, Murali TM: **The PathLinker app: Connect the dots in protein interaction networks [version 1; referees: 1 approved, 2 approved with reservations].** *F1000Res.* 2017; **6**: 58.
PubMed Abstract | Publisher Full Text | Free Full Text

3. Ono K, Muetze T, Kolishovski G, *et al.*: **CyREST: Turbocharging Cytoscape Access for External Tools via a RESTful API [version 1; referees: 2 approved].** *F1000Res.* 2015; **4**: 478.
PubMed Abstract | Publisher Full Text | Free Full Text

4. Huang LJ, Law J, Gil D, *et al.*: **Murali-group/PathLinker-Cytoscape: v1.4.1 (Version 1.4.1).** *Zenodo.* 2018.
Data Source

Open Peer Review

Current Referee Status: ✔ ✔

Version 1

Referee Report 17 July 2018

doi:10.5256/f1000research.15906.r34937

✔ **Stefan Wuchty**
Department of Computer Science, Center for Computational Science, University of Miami, Coral Gables, FL, USA

The manuscript ' Automating the PathLinker app for Cytoscape' by Huang, law and Murali introduces an API solution of their PathLinker software that was previously published. Although the PathLinker algorithm was made available through a Cytoscape app it turned out that its usage through the Cytoscape framework was initially cumbersome. With the new solution the authors used the CyREST API that allows to tap the results of the PathLinger app when used in conjunction with Cytoscape to be incorporated in user specific pipelines. As such, the authors allow through their API to use a Cytoscape app and results thus obtained to link to methods outside the Cytoscape framework. In particular, API output can easily be tapped using python and R code as well as using the Junyper notebook.

The proposed API appears very useful for researchers who want to consider the advantages of the PathFinder app w/o being constrained by the Cytoscape framework. The current manuscript describes the basic steps to get a user going. As a small change I would describe the examples in a bit more detail to show a fully worked example. In particular, Fig. 3 goes in that direction but is a bit generic. Working a real example would help with the transparency of the underlying approach.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Referee Expertise: Systems biology, bioinformatics

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Referee Report 25 June 2018

doi:10.5256/f1000research.15906.r34942

Alexander Pico
Gladstone Institutes, San Francisco, CA, USA

Identifying paths through complex networks is a common use case. The PathLinker app provides a nice approach via GUI or scripting. The methods and use cases are detailed and clear. The provided Python Notebook is especially handy.

My only suggestions for future versions of this work would be to support table column-based specification of sources and targets (already mentioned in your summary) as well as an R Markdown version of your script. The RCy3 package should make it easy to translate as it is analogous to py2cytoscape.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.