SOFTWARE TOOL ARTICLE

CySpanningTree: Minimal Spanning Tree computation in Cytoscape [version 1; peer review: 1 approved, 1 approved with reservations]

Faizaan Shaik, Srikanth Bezawada, Neena Goveas

Department of Computer Science and Information Systems, Birla Institute of Technology & Science, Goa, 403726, India

V1 First published: 05 Aug 2015, 4:476
<https://doi.org/10.12688/f1000research.6797.1>
 Latest published: 05 Aug 2015, 4:476
<https://doi.org/10.12688/f1000research.6797.1>

Abstract

Simulating graph models for real world networks is made easy using software tools like Cytoscape. In this paper, we present the open-source CySpanningTree app for Cytoscape that creates a minimal/maximal spanning tree network for a given Cytoscape network. CySpanningTree provides two historical ways for calculating a spanning tree: Prim's and Kruskal's algorithms. Minimal spanning tree discovery in a given graph is a fundamental problem with diverse applications like spanning tree network optimization protocol, cost effective design of various kinds of networks, approximation algorithm for some NP-hard problems, cluster analysis, reducing data storage in sequencing amino acids in a protein, etc. This article demonstrates the procedure for extraction of a spanning tree from complex data sets like gene expression data and world network. The article also provides an approximate solution to the traveling salesman problem with minimum spanning tree heuristic. CySpanningTree for Cytoscape 3 is available from the Cytoscape app store.

Keywords

minimum spanning tree , gene expression data , euclidean distance , Hamiltonian cycle

This article is included in the **Cytoscape** gateway.

Open Peer Review

Approval Status

	1	2
version 1		
05 Aug 2015	view	view
1. Shaillay Dogra , Vishuo BioMedical Pte Ltd, Singapore, Singapore 2. Ankush Sharma , National Research Council, Siena, Italy Institute of Clinical Physiology, Siena, Italy United Arab Emirates University, Al-Ain, United Arab Emirates		
Any reports and responses or comments on the article can be found at the end of the article.		

Corresponding authors: Faizaan Shaik (smd.faizan@gmail.com), Srikanth Bezawada (f2010451@goa.bits-pilani.ac.in)

Competing interests: No competing interests were disclosed.

Grant information: The authors declared that no grants were involved in supporting this work.

Copyright: © 2015 Shaik F *et al.* This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. Data associated with the article are available under the terms of the [Creative Commons Zero "No rights reserved" data waiver](#) (CC0 1.0 Public domain dedication).

How to cite this article: Shaik F, Bezawada S and Goveas N. *CySpanningTree: Minimal Spanning Tree computation in Cytoscape* [version 1; peer review: 1 approved, 1 approved with reservations] F1000Research 2015, 4:476 <https://doi.org/10.12688/f1000research.6797.1>

First published: 05 Aug 2015, 4:476 <https://doi.org/10.12688/f1000research.6797.1>

Introduction

Graph theory is being widely used for network analysis in various fields¹. Extraction of various kinds of subnetworks is one of the ways to identify functional modules within complex networks². A tree is a subnetwork with minimal connections. Specifically in graph theory, a tree is a graph with only one path between every two nodes. In other words, any connected graph without simple cycles is a tree. Given a connected graph, which is not a tree, one can extract a tree from it by eliminating cyclic edges. A spanning tree contains all the nodes of the graph and has $(N-1)$ edges where N is the number of nodes in the given graph. Extracting a spanning tree gets interesting when edges of the given graph have weights. In finding the minimal/maximal spanning tree, one would ideally extract the tree whose sum of weights is minimum/maximum respectively. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree. There may be several minimum spanning trees of the same weight; in particular, if all the edge weights of a given graph are the same, every spanning tree of that graph is minimal. If each edge has a distinct weight then there will be only one unique minimum spanning tree.

In this paper, we present CySpanningTree, a Cytoscape³ 3 app for extracting a spanning tree from a given graph. Once the user imports a dataset, by clicking the “Create spanning tree” button of the app, a new spanning tree network is created in the network panel of Cytoscape. Historically, spanning trees are used in various applications like constructing a road network between cities with a minimum cost, as a heuristic for the traveling salesman problem (TSP), for the spanning tree network optimization protocol in networking, clustering gene expression data, etc. Three of the mentioned cases have been demonstrated in the use cases section.

Methods

Implementation

CySpanningTree is the Java implementation of Prim's⁴ and Kruskal's algorithms⁵, using the Cytoscape 3 API and Java 7 for extracting a minimal spanning tree (MST). An MST for a given graph might not be unique, however for a given same Cytoscape session, the tie-breaking approach for selecting edges of equal weights is deterministic. The user gets the same spanning tree in a given Cytoscape session unless he reloads the network.

This tool also has a “Create Hamiltonian cycle” button which invokes the computation of the Hamiltonian cycle⁶. For computing this cycle, it first finds an MST using Prim's algorithm and then performs a pre-order traversal on it. This pre-order traversal is a modified version of the depth-first search algorithm which results in a Hamiltonian path. Later, we connect the last node and the first node of this path to make a cycle. Users are recommended to run the Hamiltonian cycle algorithm on a fully connected graph to avoid missing of the edges while traversing.

Table 1 has the complexities of the algorithms and the uniqueness of the outputs used in the app. Prim's algorithm runs using adjacency list representation of the graph and thus implemented with a complexity $O(V^2)$. Kruskal's algorithm runs using adjacency matrix of the graph and has a complexity of $O(EV^2(E+V))$. The Hamiltonian cycle first calculates a spanning tree using Prim's algorithm with a complexity of $O(V^2)$ and then runs depth-first search algorithm with a complexity $O(E+V)$.

Table 1. Comparison of algorithms used in CySpanningTree.

Algorithm	Complexity	Uniqueness
Prim's	$O(V^2)$	not unique
Kruskal's	$O(EV^2(E+V))$	not unique
Hamiltonian cycle	$O(V^2 + E)$	not unique

Graphical user interface

The GUI component of CySpanningTree is represented as a tabbed panel in the control panel of Cytoscape. Cytoscape takes care of loading the input network. The CySpanningTree menu (Figure 1) loads in the control panel of Cytoscape by selecting it from App menu. Currently the app runs only on connected networks. When the user tries to execute a spanning tree algorithm on an unconnected graph, an error message pops up. For weighted graphs, the user has to select the edge attribute from the drop down list (which is by default “None” that treats all edges with the same weight).

Setting the root node for Prim's spanning tree

Prim's algorithm starts with a root node and hence the user is asked for the same when the Prim's Spanning Tree button is pressed. If the user enters a node that is not in the network, the user gets an error message and the program terminates.

Figure 1. User interface of CySpanningTree.

Visualizations

The resultant MST or the Hamiltonian cycle network has the same layout as that of the input network with nodes positioned at the same location and edges scaled down. When spanning tree subnetworks are created, the corresponding spanning edges are highlighted in the input network. In **Figure 2**, the input network is a fully connected graph of capital cities of countries in the world, containing 203 cities and 20503 connections between them. The resultant networks: “Kruskal’s Spanning Tree”, “Prim’s Spanning Tree” and “Hamiltonian Cycle” are connected graphs containing all the 203 cities and only 202, 202 and 203 edges respectively. Spanning trees are extracted as separate Cytoscape networks under the same network collection as shown in **Figure 2**.

Network	Nodes	Edges
Capital City Network		
Input network	203(0)	20503(406)
Input network - Kruskal's Spanning Tree	203(0)	202(0)
Input network - Prim's Spanning Tree	203(0)	202(0)
Input network - Hamiltonian cycle	203(0)	203(0)

Figure 2. New networks created dynamically in Control panel.

Use cases

In this section, we present the spanning tree results on use cases with datasets in four scenarios: gene expression matrix of gene expression data, building a cost efficient road network when all possible costs are known, an approximate solution to the travelling salesman problem and connecting a 10-home village with phone lines with minimum wiring. In each scenario, the contents of the network are introduced first and then extraction of spanning trees is demonstrated.

MST of gene expression data

The expression levels of genes when exposed to various environmental conditions are recorded at different times with different samples. This data is called gene expression data and is analyzed to extract the similarities between genes. Gene expression data $G(\vec{g}_1, \vec{g}_2, \dots, \vec{g}_n)$ for n genes is multi-dimensional data with each $\vec{g}_i = (d_i^1, d_i^2, \dots, d_i^m)$ for given m expression levels. Here $\vec{g}_i$ represents the i^{th} gene and d_i^j represents the j^{th} expression level of this i^{th} gene.

$$G = \begin{bmatrix} d_1^1 & d_1^2 & d_1^3 & \dots & d_1^m \\ d_2^1 & d_2^2 & d_2^3 & \dots & d_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_n^1 & d_n^2 & d_n^3 & \dots & d_n^m \end{bmatrix}$$

This data has been simulated as a graph with nodes being genes and edges being the genetic distance between them. Genetic distance is defined as the measurement of similarity between genes.

$$\text{Euclidean distance between genes } \vec{g}_i \text{ and } \vec{g}_j = \sqrt{(d_i^1 - d_j^1)^2 + (d_i^2 - d_j^2)^2 + \dots + (d_i^m - d_j^m)^2}$$

For each pair of genes, this genetic distance is calculated which gives a fully connected graph. The data set⁷ has been taken from the *Saccharomyces* Genome Database and contains expression levels of budding yeast — *S. cerevisiae* with a total of 6149 genes (http://downloads.yeastgenome.org/expression/microarray/Cho_1998_PMD_9702192/). Typically, it becomes difficult to visualize a large graph of 6149 nodes with each node connected to every other node in the graph. A spanning tree of the gene expression data makes it possible to visualize such a large network as shown in **Figure 3**.

- Input network: A fully connected graph of *S. cerevisiae* expression data
- Nodes: Genes of *S. cerevisiae*
- Edges: Euclidean distance between genes calculated using expression levels
- Output network (**Figure 3**): Kruskal’s spanning tree of the input gene expression data

Although a lot of edges are removed from the network during the process of creating a spanning tree, no essential information is lost⁸. A spanning tree is a better way to visualize large networks compared to fully connected graphs. We observed that genes with similar functionalities are connected closely in the resultant spanning tree. Many clustering algorithms have been applied to gene expression data^{8,9}, we are currently working on clustering using minimum spanning trees for our next release of CySpanningTree.

Figure 3. Spanning tree obtained from graph of *S. cerevisiae* expression data; Layout: Allegro Spring-Electric layout using Allegro Layout app in Cytoscape.

MST on world network

This dataset¹⁰ consists of nodes which are capital cities of all countries in the world and edges between them representing the distance in kilometers. These distances are measured using latitude and longitude coordinates of the cities (<http://privatewww.essex.ac.uk/~ksg/data-5.html>). This dataset, when imported into Cytoscape, results in a fully connected graph as the distance is calculated for each pair of capital cities. Prim's algorithm has been executed on this dataset to produce a MST network as shown in Figure 5

- Input network: Fully connected graph of capitals cities as shown in Figure 4
- Nodes: Capital cities of all countries in the world
- Edges: Displacement between cities
- Output minimum spanning tree: Network with minimum cost such that each city is connected. Cities separated with large distances are represented with strong edges as shown in Figure 5

Furthermore, this solution can be used for drawing a Hamiltonian cycle which is an approximation to the Travelling Salesman problem. Drawing a Hamiltonian cycle for a smaller network is discussed in the next subsection.

MST as a heuristic solution for the TSP

The TSP is a well-known combinatorial optimization problem. The goal is to find the shortest tour that visits each city in a given list exactly once and returns to the starting city. Though the

Figure 4. Fully connected graph of the capital city network; Layout: Allegro Spring-Electric layout using Allegro Layout app in Cytoscape.

Figure 5. Minimum Spanning Tree of the capital city network; Layout: Allegro Spring-Electric layout using Allegro Layout app in Cytoscape.

problem statement looks simple, TSP is NP-complete¹¹. Even though the problem is computationally difficult, a large number of heuristic solutions¹² are known due to the number of applications of this problem¹³ like planning, logistics, DNA sequencing, predicting protein functions, etc.

Pre-order traversal on a minimum spanning tree is one of the heuristic solutions for TSP^{5,14}. In this subsection, a Hamiltonian cycle is drawn for a spanning tree to show that the resultant cycle is a near solution to the TSP. The optimal TSP tour in Figure 9 is about 17% shorter than the Hamiltonian cycle obtained using spanning tree in Figure 8. On executing the Hamiltonian cycle algorithm on the input network, the software will create both Prim's spanning tree as well as the Hamiltonian cycle. Five nodes from the above capital city network are used for the TSP use case.

- Input network: Fully connected graph of 5 capital cities
- Nodes: Capital cities of countries: USA, Brazil, South Africa, India and Italy
- Edges: Displacement between cities shown in kilometers

Connecting a 10-home village with phone lines

This dataset consists of houses depicted as nodes and the edges are the means by which one house can be wired up to another. The weights of the edges dictate the distance between the houses. The task of the telephone company is to wire all houses using the least amount of telephone wiring possible.

- Input network: Houses in village depicted as graph as shown in Figure 10

Figure 6. Fully connected graph of 5 cities and their displacements.

Figure 8. Hamiltonian cycle drawn from the spanning tree with USA as starting node.

Figure 7. MST of the network in Figure 6.

Figure 9. Optimal TSP tour from USA.

Figure 10. Houses depicted as nodes.

- Nodes: Houses H_1 to H_{10}
- Edges: Distance between the houses
- Output MST: Network which connects the houses via wires with least possible wiring. **Figure 11** and **Figure 12** are the spanning trees obtained using Prim's (H1 as root node) and Kruskal's algorithm, respectively.

Figure 11. MST using Prim's algorithm.

Figure 12. MST using Kruskal's algorithm.

Summary

In this paper, we present CySpanningTree app for Cytoscape 3. CySpanningTree fills an important need for many Cytoscape users and researchers in obtaining spanning trees across different types of networks. CySpanningTree makes effective use of the Cytoscape 3 API in extracting the subnetwork and creating it as a separate network. In the near future, we will be exploring MST based clustering and we are determined to explore more datasets whose spanning tree evaluation is significant.

Software availability

CySpanningTree app can be downloaded from the Cytoscape app store.

Software available from

<http://apps.cytoscape.org/apps/cyspanningtree>

Latest source code

<https://github.com/smd-faizan/CySpanningTree>

Archived source code as at the time of publication

<http://dx.doi.org/10.5281/zenodo.1966815>

Licence: Lesser GNU Public License 3.0

<https://www.gnu.org/licenses/lgpl.html>

Author contributions

FS and SB conceived the CySpanningTree app. NG supervised the project. FS contributed to the implementation of Kruskal's algorithm, Hamiltonian cycle and user interface of the app. SB contributed to the implementation of Prim's algorithm. FS and SB worked on the use cases. FS and SB wrote the manuscript. NG participated in the design of the app and in the revision of the manuscript.

Competing interests

No competing interests were disclosed.

Grant information

The author(s) declared that no grants were involved in supporting this work.

Acknowledgments

The authors would like to thank their professor Bharat.M.Deshpande for shaping and motivating their interests towards Discrete Mathematics, Scooter Morris from Cytoscape open source community for helping with Cytoscape API to extract the subnetwork in an intuitive way.

Supplementary material

Cytoscape session files for use cases. Cytoscape session files (*.cys) for the TSP, world network, and 10-home village use cases.

[Click here to access the data.](#)

References

1. Pavlopoulos GA, Secrier M, Moschopoulos CN, *et al.*: **Using graph theory to analyze biological networks**. *BioData Min.* 2011; **4**(10): 1–27.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
2. Lemetre C, Zhang Q, Zhang ZD: **SubNet: a Java application for subnetwork extraction**. *Bioinformatics.* 2013; **29**(19): 2509–11.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
3. Shannon P, Markiel A, Ozier O, *et al.*: **Cytoscape: a software environment for integrated models of biomolecular interaction networks**. *Genome Res.* 2003; **13**(11): 2498–2504.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
4. Prim RC: **Shortest connection networks and some generalizations**. *Bell System Technical Journal.* 1957; **36**(6): 1389–1401.
[Publisher Full Text](#)
5. Kruskal JB: **On the shortest spanning subtree of a graph and the traveling salesman problem**. *Proc Am Math Soc.* 1956; **7**(1): 48–50.
[Publisher Full Text](#)
6. West DB, *et al.*: **Introduction to graph theory**, volume 2. Prentice hall Upper Saddle River. 2001.
[Reference Source](#)
7. Cho RJ, Campbell MJ, Winzeler EA, *et al.*: **A genome-wide transcriptional analysis of the mitotic cell cycle**. *Mol Cell.* 1998; **2**(1): 65–73.
[PubMed Abstract](#) | [Publisher Full Text](#)
8. Xu Y, Olman V, Xu D: **Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees**. *Bioinformatics.* 2002; **18**(4): 536–545.
[PubMed Abstract](#) | [Publisher Full Text](#)
9. Jiang D, Tang C, Zhang A: **Cluster analysis for gene expression data: a survey**. *IEEE Trans Knowl Data Eng.* 2004; **16**(11): 1370–1386.
[Publisher Full Text](#)
10. Gleditsch KS: **Distance between capital cities**. 2008.
[Reference Source](#)
11. Papadimitriou CH: **The Euclidean travelling salesman problem is NP-complete**. *Theor Comput Sci.* 1977; **4**(3): 237–244.
[Publisher Full Text](#)
12. Rosenkrantz DJ, Stearns RE, Lewis PM II: **An analysis of several heuristics for the traveling salesman problem**. *SIAM J Comput.* 1977; **6**(3): 563–581.
[Publisher Full Text](#)
13. Lenstra JK, Rinnooy Kan AHG: **Some simple applications of the travelling salesman problem**. *J Oper Res Soc.* 1975; **26**: 717–733.
[Publisher Full Text](#)
14. Held M, Karp RM: **The traveling-salesman problem and minimum spanning trees**. *Operations Research.* 1970; **18**(6): 1138–1162.
[Publisher Full Text](#)
15. Shaik F, Bezawada S: **CySpanningTree: Hamiltonian**. *Zenodo.* 2015.
[Data Source](#)

Open Peer Review

Current Peer Review Status:

Version 1

Reviewer Report 29 March 2016

<https://doi.org/10.5256/f1000research.7304.r12115>

© 2016 Sharma A. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Ankush Sharma

¹ Institute of Clinical Physiology, National Research Council, Siena, Italy

² LISM, Institute of Clinical Physiology, Siena, Italy

³ Faculty of Information Technology, United Arab Emirates University, Al-Ain, United Arab Emirates

In this research article entitled -"CySpanningTree: Minimal Spanning Tree computation in Cytoscape, the authors describe the *app* for Cytoscape version 3 that creates minimal/maximal spanning tree for a given network using network Prim's and Kruskal's algorithms. The *CySpanningTree* app appears to be useful in approximating the minimum-cost weighted perfect matching, maximum flow problems and other related issues (Supowit et al. 1980; Dahlhaus et al. 2006). The description of the proposed implementation of *CySpanningTree* app for Cytoscape version 3 is informative and detailed for audience. The article provides sufficient details with appropriate title and well-written abstract.

Minor Concerns

1. Some more details on usage on practical applications are strongly suggested to include in this research article as requested by Reviewer 1 in Point 2.
2. The definition of gene expression and generalizing gene expression data in one context is not correct in section MST of gene expression data. It is highly recommended to correct it and cite appropriate research articles defining gene expression and Gene expression data.
3. Gene-gene interaction network reconstruction from gene expression needs to be detailed in methodology sections e.g. how edge weights are calculated and then used for calculation of Euclidean distance between genes.
4. The usage of Genetic distance seems to be inappropriate in this context as it is a measure of the genetic divergence between species or between populations within a species. Please elaborate, if it is used in this context in research article.
5. I would suggest making comprehensive figures for better readability e.g. (figure 1 and

figure 2 may be merged into figure 1, Similarly figure 4,5,6,7 into figure 3, figure 8, 9 into figure 4 and figure 10, 11, 12 into figure 5) and brief description of figures in text as well as in legend will make help in better understanding of the examples and usage of the cySpanning trees.

References

1. Dahlhaus E, Johnson D, Papadimitriou C, Seymour P, et al.: The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*. 1994; **23** (4): 864-894 [Publisher Full Text](#)
2. Supowit KJ, Plaisted DA, Reingold EM: Heuristics for weighted perfect matching. *Proceedings of the twelfth annual ACM symposium on theory of computing – STOC '80*. 1980; **New York, New York, USA: ACM Press**: 398-419

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 14 September 2015

<https://doi.org/10.5256/f1000research.7304.r10256>

© 2015 Dogra S. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Shaillay Dogra

Vishuo BioMedical Pte Ltd, Singapore, Singapore

The authors have come up with a useful plug-in for cytoscape. Different algorithms have been implemented to reduced a cluttered network to a more meaningful one. Such efforts are welcome and potentially useful especially for those working in network analysis and visualization.

The manuscript can be enhanced by considering the suggestions below:

1. Include a schematic figure to illustrate the points mentioned in the Introduction for the benefit of a wider audience or non specialist users like experimental biologists.
2. It will be helpful to intended users like experimental biologists if different algorithm choices were explained in terms of what they mean, in which case it is advised to use which particular algorithm etc.
3. The author's mention that different sessions may lead to different trees. What are the potential pitfalls of this in generating results and possible different interpretations. Please discuss this aspect.

4. How do the authors define genetic distance? It is not clear. Is it based on correlation value of expression of genes? Please elaborate.

5. Figure 5, "MST on world network" - how to use a weight; for ex., 'effective distance' between cities that is a measure of air-connectivity can be used to depict 'realistic distance' than physical distance.

6. More discussion on interpretation of figures 6,7 and figures 8,9 will be helpful to the readers.

7. What is a way to verify that the solution is actually what it is 'supposed to be'?

Competing Interests: No competing interests were disclosed.

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

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research