Naval Research Laboratory

Washington, DC 20375-5320

NRL/MR/5580--18-9767

Node Ranking Tool - NoRT

Ira S. Moskowitz
Jonathan W. Decker

Information Management & Decision Architectures Branch
Information Technology Division

Rebekah H. Kang

University of Virginia
Charlottesville, VA

March 23, 2018

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
23-03-2018	Memorandum Report	October 2015 - September 2017

4. TITLE AND SUBTITLE

Node Ranking Tool - NoRT

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62235N

6. AUTHOR(S)

Ira S. Moskowitz, Jonathan W. Decker and Rebekah H. Kang*

5d. PROJECT NUMBER

5e. TASK NUMBER
IT-235-018

5f. WORK UNIT NUMBER
6804

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/5580--18-0039

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
One Liberty Center
875 North Randolph Street, Suite 1425
Arlington, VA 22203-1995

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

* University of Virginia, Charlottesville, VA author's email: ira.moskowitz@nrl.navy.mil

14. ABSTRACT

This paper gives a description of the Node Ranking Tool (NoRT) that was developed as part of the Applied Network Science 6.2 base program work unit at NRL, Code 5580. We explain the theory of NoRT and how to use it.

15. SUBJECT TERMS

TOPSIS, Social Network, Sensor Network, Centrality, Diffusion, Disease, Virus, Expectation, Pandemic, Closeness, Graph, Degree, Spectrum

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Ira S. Moskowitz
Unclassified Unlimited	Unclassified Unlimited	Unclassified Unlimited	SAR	23	**19b. TELEPHONE NUMBER** *(include area code)* (202) 404-7930

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

This page intentionally left blank.

U.S. NAVAL RESEARCH LABORATORY

Node Ranking Tool—NoRT

Ira S. Moskowitz*, Jonathan Decker§, Rebekah Kang¶

*Information Management & Decision Architectures Branch
Code 5580, Naval Research Laboratory
Washington, DC 20375

§Information Management & Decision Architectures Branch
Code 5581, Naval Research Laboratory
Washington, DC 20375

¶ NRL Code 5580, NREIP Summer Intern 2017
University of Virginia
Charlottesville, VA 22903

Abstract

This paper describes the *Node Ranking Tool* (NoRT) that was developed as part of the Applied Network Science 6.2 base program work unit at NRL. We explain the theory of NoRT and how to use it.

Index Terms

TOPSIS, Social Network, Sensor Network, Centrality, Diffusion, Disease, Virus, Expectation, Pandemic, Closeness, Graph, Degree, Spectrum.

I. Introduction

This paper gives a description of the *Node Ranking Tool* (NoRT) that was developed as part of the Applied Network Science 6.2 base program work unit at NRL, Code 5580. We explain the theory of NoRT and how to use it. Examples are supplied. We first explain centrality, then we explore some applications, and then we turn to a discussion of NoRT itself.

We wish to determine which nodes in a network are "strong" and which are "weak". We model a network as a connected undirected graph G with a set of nodes $N = \{\nu_1, ..., \nu_m\}$. G has as an adjacency matrix A, which is a symmetric matrix that has i, j entry 1, if and only if there is a link between ν_i and ν_j, and is zero otherwise. We denote the link, if it exists, between ν_i and ν_j as $l_{i,j} = 1$. If there is no link, then $l_{i,j} = 0$. The set of $\{l_{i,j}\}$ is denoted by L. Note that all $l_{i,j}$ have a constant weight of 1 or 0, and there are no self-loops. Hence, as noted A is symmetric with zeros down the diagonal and 0's or 1's as the off diagonal entries, where $A_{i,j} = \text{weight}(l_{i,j})$.

In this paper, the network and the graph are taken as identical entities. We envision an attack on the network in the form of nodes becoming "compromised". Once a node is compromised, it acts as an "agent" [25] of the attack and attempts to compromise other nodes. We wish to rank the network nodes in terms of their ability to compromise other parts of the network, or to be compromised themselves.

In our view, a network being compromised via the spread of a "disease" [9]. Once a disease is in a node(s), we are concerned with how the disease spreads to the rest of the network. Of interest is the asymptotic behavior of the disease, particularly in the way it affects the other nodes in the long term. That is, do nodes become healthy once they are infected? Do all the nodes eventually become infected resulting in a pandemic? Do certain parts of the network become infected, while others are left untouched? We may also be concerned with how a disease spreads over a short period of time.

Manuscript approved November 17, 2017.

We use a combination of various forms of graph centrality to determine the likelihood of a node to compromise other parts of the graph, or to become compromised itself. Furthermore, we also consider the asset criticality [17] of the graph nodes. That is, some nodes are more important than others in terms of becoming infected, or spreading disease. Our study of disease propagation must therefore be fine-tuned to consider asset criticality, which is important if we wish to plan defensive or offensive measures involving certain specific nodes.

We balance all of the above concepts by using the ideas as given in [7, 12, 18] (which discusses Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) which is a Multi-Attribute Decision Making (MADM) technique), node importance, and asset criticality. These concepts will be made clear in the rest of the document.

I-A. What is NoRT?

NoRT is an application of the TOPSIS method where there are four criteria, consisting of the different node centralities discussed below. Furthermore, NoRT has a fifth optional criteria called asset criticality which allows us to individually denote nodes that are more important than others.

NoRT is a self-contained SageMath script run from the command line. The code is included in this paper.

Experiments have been run to validate the theoretical basis of using NoRT. Extensions to the present NoRT code are easily done.

II. Node Centrality

There are many different measure of node centrality and importance [26, Ch. 7]. We concentrate on four in NoRT. They are (1) *Degree Centrality* (DC), (2) *Closeness Centrality* (CC), (3) *Betweenness Centrality* (BC), and (4) *Eigenvector Centrality* (EC).

Every connected graph has a natural distance function for its nodes. We say that the distance between ν_i and ν_j is given by

$$d(\nu_i, \nu_j) := \text{shortest (in terms of the number of links) path length between } \nu_i \text{ and } \nu_j. \tag{1}$$

Given ν_i and ν_j, we say that a path between ν_i and ν_j is a *geodesic* iff the length of that path is $d(\nu_i, \nu_j)$. Note that a geodesic between ν_i and ν_j need not be unique. We denote the set of geodesics between ν_i and ν_j as $G(\nu_i, \nu_j)$. We see that for the 3E graph in Fig. 1 that $G(1, 4) = \{(1\text{-}0, 0\text{-}4), (1\text{-}3, 3\text{-}4)\}$, a set with two paths.

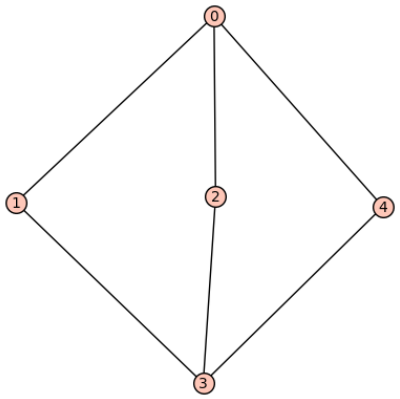

Fig. 1: 3E Graph

However, $G(0, 3)$ is made up of three geodesics paths $\{(0\text{-}1, 1\text{-}3), (0\text{-}2, 2\text{-}3), (0\text{-}4, 4\text{-}3)\}$.

II-A. Degree Centrality (DC)

DC is simply the normalized[1] degree (the number of links into a node) of a node. For node ν_i we have

$$DC(\nu_i) := \frac{1}{m-1} \sum_j A_{i,j} \ . \tag{2}$$

Since our graph is connected, DC can never be zero, and the most links a node can have is $m-1$. Therefore $DC(\nu_i)$ ranges from $\frac{1}{m-1}$ to 1, where m is the cardinality of the set of nodes.

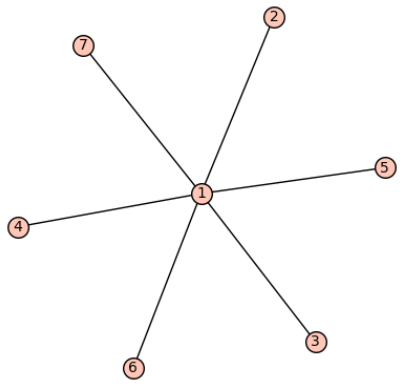

Fig. 2: Star graph 7 nodes

In Fig. 2 we see that every node is connected to ν_1, which has maximal $DC = 1$, and all the other nodes have $DC = 1/6$.

DC is a reliable measure for how quickly one node can send "activity" out over a link and compromise other nodes, especially if the compromising ability weakens over time, or is probabilisitic in nature. That is, one can view DC as measuring how a diffusive wave spreads over time from a node, with its amplitude diminishing over time/hops.

II-B. Closeness Centrality (CC)

For node ν_i, we consider $\sum_j d(\nu_i, \nu_j)$, take its inverse, and normalize by $m-1$. Again, we note that the normalization is optional; however, we use the same algorithms as SageMath [5]. In general, our centrality measures may differ from others in the literature by constants. This does not affect node rankings so is insignificant to us.

The closeness centrality (CC) of node ν_i is given as

$$CC(\nu_i) := \frac{m-1}{\sum_j d(\nu_i, \nu_j)} \ . \tag{3}$$

[1]When we introduce the TOPSIS we will see that how attribute values are normalized is irrelevant.

Thus for Fig. 2 we have that $CC(\nu_1) = 1$ and $CC(\nu_j) = \frac{6}{2\cdot 5+1\cdot 1} = 6/11,$ for $j = 2, ..., 7$.

Centrality closeness basically tells us that a node that is closer to the other nodes has a larger CC than one that is further away. A good discussion of CC can be found in [26, 7.6]. Closeness centrality gives us a measure of how quickly a disease may spread from one node to the rest of the nodes.

II-C. Betweenness Centrality (BC)

Another important measure of centrality is BC. Given node ν_i, we consider two subsets of all the geodesics of G. The first is simply $G(\nu_j, \nu_k)$, as defined above, and the other is $G|_{\nu_i}(\nu_j, \nu_k)$, which is the subset of $G(\nu_j, \nu_k)$ consisting of all geodesics between ν_j and ν_k, provided that ν_i is an interior (not end) point of the geodesic. We define the Betweenness centrality as:

$$BC(\nu_i) := \binom{m-1}{2}^{-1} \cdot \sum_{j\neq i\neq k} \frac{\|G|_{\nu_i}(\nu_j, \nu_k)\|}{\|G(\nu_j, \nu_k)\|} \ . \tag{4}$$

A high BC value is indicative of being a very critical node in terms of being an intermediary for information transmission.

Looking at Fig. 2, we see that node 1 is interior to every geodesic between the other nodes. We only have one geodesic between any two nodes 2,...,7, since a geodesic does not care about direction, this gives us 15 geodesics without node 1 as an endpoint. However, each one of these geodesics goes through node 1. Hence

$$BC(1) = \binom{6}{2}^{-1} \cdot 15 = 1$$

which makes the strange normalization term clear. But on the other hand $BC(j) = 0, j = 2, ..., 7$.

However, node $j, j = 2, ..., 7$ is never interior to any geodesic in G, so $G|_{\nu_j}(\nu_i, \nu_k) = \emptyset; i, k$ arbitrary, $j = 2, ..., 7$. Therefore, $BC(j) = 0, j = 2, ...7$ in Fig. 2.

Now looking at Fig. 1, we calculate $BC(1)$. The only subset of the geodesics that contains node 1 as an interior point is $G(0, 3)$. In this subset there is only one geodesic that has node 1 as an interior point. Since $\binom{5-1}{2}^{-1} = 1/6$, we have that $BC(1) = 1/18$. Appealing to symmetry, we also see that $BC(2) = BC(4) = 1$. Now consider $BC(0)$, the subsets that have node 0 as interior points are $G(1, 4)$, $G(1, 2)$ and $G(2, 4)$. Since each of these subsets has two elements, with only one element per subset having node 0 as an interior point, we have that $BC(0) = 1/6 \cdot (1/2 + 1/2 + 1/2) = 1/4$. Again, by symmetry $BC(3) = 1/4$ also.

II-D. Eigenvector Centrality (EC)

Our final, and extremely important, measure of centrality is EC. Note there are many other measures of centrality. We feel that the four in this document suffice for most network vulnerability analyses. Additional measures can easily be added into NoRT, if need be.

The adjacency matrix A of G is obviously non-negative. Therefore we have the following famous result [19, p. 536, Thm,1] and [19, p. 543, Thm,1] .

Theorem 2.1: (Perron-Frobenius) A has a real eigenvalue λ_{max} greater than or equal to the magnitude of any of its other eigenvalues[2]. There is a nonnegative[3] eigenvector $\mathfrak{v}_A$ associated with λ_{max}.

To settle the choice of normalization once and for all, we use the *Gould vector* [10], $\mathfrak{g}$, as the unique representative of $\mathfrak{v}_A$ with Euclidean length 1. We use the notation $\hat{u_i}$ to denote the unit vector that is 0 in all components, except for the ith component which is 1. We define [2],[26, Ch. 7] the Eigenvalue centrality (EC) of node ν_i, to simply be the i component of $\mathfrak{g}_{\mathbb{A}}$,

$$EC(\nu_i) := \mathfrak{g} \cdot \hat{u_i} \ . \tag{5}$$

[2]The spectral radius of A is also called λ_{max}.

[3]Of course, $\mathfrak{v}_a$ is unique only up to a positive scalar.

The spectral radius is used as a phase transition point for various models in virology. The relative size of the components of the Gould vector influences the spectral radius. What this implies is that the most effective place to put a disease, in terms of infiltration, is in the node that has the largest component value of the Gould index, see [28, 23] and Shield value [3]. EC is also a key part of many page rank algorithms.

For Fig. 2 we have that $\lambda_{max} = \sqrt{6}$, and $\mathfrak{g} = \frac{1}{\sqrt{2}} \cdot \langle 1, 1/\sqrt{6}, 1/\sqrt{6}, 1/\sqrt{6}, 1/\sqrt{6}, 1/\sqrt{6}, 1/\sqrt{6} \rangle$. Not surprisingly the node (1) in the middle has the highest EC value.

II-E. Interesting Example 1: the Kite

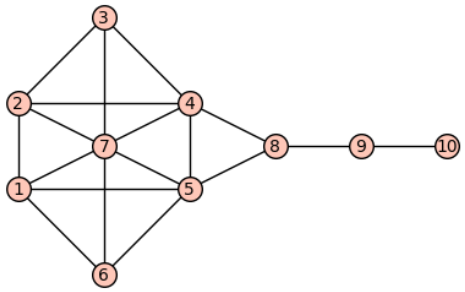

Fig. 3: Kite: Interesting example from [6, Ex. 1]

Consider Fig. 3. Which is the correct metric of centrality? Let us make a table of the different centrality measures with the node ranking next to the values.

We see that the node ranking/importance is very much dependent upon which centrality measure we use. Therefore, following [6] and [3], we use TOPSIS to evaluate our decisions about the relative importance of the nodes in a graph. Of course, in this example, DC and EC display the same ranking, but for other graphs that need not be the case. However, the raw values of these ranks affects the TOPSIS process. For example, in Fig. 4 the six unique (highest to lowest) EC rankings are nodes 1,5,6,4,3,2, and the three DC rankings are 1,5,(3,4,6),2. Where () means they are the same ranking.

Node in Rank Order	DC*9 Value
7	6
4	5
5	5
1	4
2	4
3	3
6	3
8	3
9	2
10	1

Node in Rank Order	CC Value
4	0.64
5	0.64
7	0.6
8	0.6
1	0.53
2	0.53
3	0.5
6	0.5
9	0.43
10	0.31

Node in Rank Order	BC Value
8	0.39
4	0.23
5	0.23
9	0.22
7	0.1
1	0.02
2	0.02
3	0
6	0
10	0

Node in Rank Order	EC Value
7	0.48
4	0.4
5	0.4
1	0.35
2	0.35
3	0.29
6	0.29
8	0.2
9	0.05
10	0.01

TABLE I: Rankings against Four Different Centrality Measures for the Kite Fig. 3. Note that BC is normalized by $\left(\frac{N}{2} \cdot \frac{N-1}{2}\right)^{-1}$ instead of $\binom{N}{2}^{-1}$ to agree with [6] (this has no effect on the rankings).

II-F. Questions

If we were to "attack" the network, which node would we attack first? If we were to defend the network, where would we put the most defensive firewalls? If we were to start removing links to thwart an attack, at which node(s) would we remove the links? If some nodes are more critical than others, how would that influence the above? We will show later in this paper how TOPSIS can answer these questions.

III. Social Networks

We quote from [24]

> Social networks are a core part of modern information sharing and communication and distinctly related to Internet media consumption. As a result, understanding information flow within these networks is of great interest to both industry and government [20]. Social networks are used by roughly 20% of the people in the United States to make decisions [22], which is a direct indication of the importance of social networks such as Facebook, Twitter, etc. Within this context, the transmission of information across the social network topology can provide indications of events and situational awareness of all sorts, e.g. terrorist activities, marketing penetration, social consciousness, etc. [1]. Thus researchers have applied many techniques to understand information transmission/diffusion in the characterization of this phenomenon. The diffusion of innovation over a network is one of the original reasons for studying networks. Furthermore, the spread of disease among a population has been studied for centuries [11].
>
> The basis for many of the social network analysis tools that seek to understand or predict diffusion is the concept of centrality [29, 21, 4]. In most applications of social network analysis, centrality typically is used to identify a node/vertex that is the most influential or having the most importance. In the case of flow, spread, or diffusion, this can be taken as a super spreader [2]. For brevity, we begin by

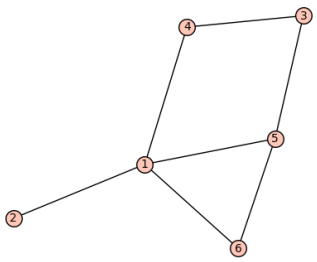

Fig. 4: EC and DC rank differ

highlighting the importance and utility of centrality to social network analysis. A thorough discussion of centrality measures, as well as some of their limitations, can be found in Freeman's work [8]. Freeman correctly pointed out that centrality must be considered in the context of the network structure/topology and offers centrality as control, centrality as independence, or centrality as activity. Relative to social network information diffusion, and our work here, we consider centrality in all three contexts and focus on a significant consideration in its use within that domain.

Well-studied metrics of centrality [26, Ch. 7] do not take probability into account. In [24] some surprising examples were shown emphasizing that a study of graph behavior based solely on topological and degree properties is incomplete when it comes to modeling virus or information propagation. We emphasize the spectral radius of a connected graph because when there is both an infection/information transmission and cure rate, the inverse spectral radius gives one a phase transition. Furthermore, the eigenvector corresponding to the spectral radius gives us a very important measure of node centrality. However, we de-emphasize the cure rate, which is consistent with information transmission such as might occur within a social network (this paragraph paraphrased from [24]).

In [20] a detailed study of "influencers" is given. They also discuss the interesting inverse problem of determining where a social influence started and potentially controlling mis-information. They discuss three information diffusion models in social networks and how the centrality measures of degree, betweenness, closeness, and eigenvalue type centrality must be analyzed in different venues of social network analysis. In [20, Sec. 2.2.2] they discuss how different measure of centrality for identifying influencer nodes are discussed in the literature.

In [21] different measures of centrality are analyzed with respect to media coverage of actors in social media. In [15, Sec. 2.2.4] measures of centrality similar to what we use are discussed in detail and compared.

IV. Sensor Networks

In [14] the authors discuss the need for more work on centrality measures for sensor networks. The same authors show further results with respect to eigenvector centrality and randomly deployed sensor networks in [16].

We note that temporal changes in sensor networks can be studied using the TOPSIS method we propose. For example, assume that we have sensors that are weather dependent. That is, as a storm front moves into range, the strength of the information reported by various sensors/nodes can deteriorate, and then return to normal after the storm passes. This can be modeled by using weights that are time varying in NoRT.

V. TOPSIS & NoRT

TOPSIS was first discussed in [13]. Our two main sources are given above, and [27].

1) We start with a $m \times n$ Decision Matrix $D = d_{m,n}$, where the rows $i = 1, ..., m$ are the alternatives (e.g. nodes), and the columns $j = 1, ..., n$ are the criteria (e.g. centrality measures, node asset values). The alternatives are $\{\mathcal{A}_i\}$, and the criteria are $\{\mathcal{K}_j\}$. The set of criteria that is indexed by j, consists of both "benefit" criteria $\mathcal{K}^+$ and "cost" criteria $\mathcal{K}^-$. If there are no cost criteria, the TOPSIS algorithm simplifies. For NoRT we are able to restrict to only benefit criteria. This is because any cost effects can be incorporated into the non-centrality criteria called asset criticality (AS). The higher the AS value is of a node the more important that node is to us. Therefore, with respect to NoRT, $\{\mathcal{K}_j\} = \mathcal{K}^+$, and $\mathcal{K}^- = \emptyset$.
   Hence, we *simplify notation* and use $\mathcal{A}$ to denote the set alternatives, and $\mathcal{K}$ to denote the set of criteria.
2) We next form a normalized decision matrix $\bar{D} = \delta_{i,j}$ by normalizing every entry in D_j, the jth column of D, by its column norm $\|D_j\|$.

$$\delta_{i,j} = \frac{d_{i,j}}{\sqrt{\sum_{i=1}^{m}(d_{i,j})^2}}, \quad i = 1, ..., m; j = 1, ..., n. \tag{6}$$

   We note that if D_k is already normalized (as in the Gould vector for Eigenvalue centrality) that $\delta_{i,k} = d_{i,k}$.
3) We now assign a weight of $0 \leq w_j$ to each column and form the weighted normalized decision matrix $V = v_{i,j}$

$$v_{i,j} = w_j \cdot \delta_{i,j} \quad i = 1, ..., m; j = 1, ..., n. \tag{7}$$

   Even though it is not necessary we normalize the weights so that $\sum_j w_j = n$.
4) We next calculate the positive ideal solution A^+ and the negative ideal solution A^-.

$$A^+ = \{v_1^+, ..., v_n^+\};, \text{ and} \tag{8}$$
$$A^- = \{v_1^-, ..., v_n^-\}, \text{ where} \tag{9}$$

$$v_j^+ = \max_i v_{i,j}, \text{ and} \tag{10}$$
$$v_j^- = \min_i v_{i,j}, \quad j = 1, ..., n. \tag{11}$$

   Thus, A^+ corresponds to finding the largest value in every criteria (column), and A^- to the minimal.
5) Now we go through attribute by attribute (row) to see how the actual values compare to the positive ideal, and the negative ideal. We calculate the Positive (Negative) Separation S_i^+ (S_i^-) between each alternative and the positive ideal solution (negative ideal solution).

$$S_i^+ = \sqrt{\sum_{j=1}^{n}(v_{i,j} - v_j^+)^2}, \text{ and} \tag{12}$$

$$S_i^- = \sqrt{\sum_{j=1}^{n}(v_{i,j} - v_j^-)^2}, \quad j = 1, ..., n. \tag{13}$$

6) We next calculate the relative closeness of alternate i to the ideal positive solution. We want to be close to the ideal positive solution and far from the ideal negative solution (see [30]). The relative closeness of the i alternate (node for us) is defined as

$$C_i = \frac{S_i^-}{S_i^- + S_i^+}. \tag{14}$$

   When $C_i = 1$ ($C_i = 0$) alternative i is the best (worst) solution, that is it coincides with A^+ (A^-).
   Note that if we have a weighting w_j and replace it with $K \cdot w_j$ the terms S_i^+ and S_i^- are multiplied by K, but the C_i values are unchanged. In particular if every column is equi-weighted at 1, and we change the normalization to $w_j = 1/n$ the C_i are unchanged. We choose the default normalization of $w_j = 1$ if all weights are equal so that we can ignore step (2) above.
7) Now we rank the alternatives, from highest to lowest, via the C_i value.

V-A. Example 2

For completeness, we include the following example for algorithm verification against a published example of TOPSIS. We note that this example has nothing to do with graphs or nodes. The alternatives are years 2008, ..., 2012; $i = 1, ..., 5$)) and criteria (various rates SR,PR, and CR, $j = 1, 2, 3$. We use a weighting of $w_j = 1/3$ for each criteria. We see that the rankings of the years in descending order are 2010, 2012, 2011, 2008, 2009, as shown by our excel calculations below (and agreeing with [6]).

YEAR	SR	PR	CR	SRn=vi1	PRn=vi2	CRn=vi3	A+ =vj+	A- = vj-
2008	0.950	0.953	0.950	0.144	0.148	0.153	0.152	0.144
2009	1.000	0.900	0.902	0.152	0.140	0.145	0.153	0.140
2010	0.974	0.975	0.946	0.148	0.152	0.152	0.153	0.145
2011	0.984	0.982	0.903	0.149	0.153	0.145		
2012	1.000	0.974	0.925	0.152	0.152	0.149		
	S+		S-		Ci		RANK	
	S1+	0.009	S1-	0.011	C1	0.561	4.000	
	S2+	0.015	S2-	0.008	C2	0.337	5.000	
	S3+	0.004	S3-	0.014	C3	0.773	1.000	
	S4+	0.008	S4-	0.014	C4	0.634	3.000	
	S5+	0.004	S5-	0.014	C5	0.772	2.000	

Fig. 5: Example 2 (Table 2) from [6]

.

V-B. Return to Interesting Example 1

Once again, we consider the Kite as shown in Fig. 3. This time we will use the normalizations that we discussed at the beginning of the paper, and we will add a column for asset criticality (AC). We start with a default value of 1 for every row in AC. Of course this node ranking agrees with the ranking done without any asset criticality.

	#	X DC	Closeness	Between	EC	Squares and sums					R				S+	S-	C		
Beverly	1	0.4444	0.059	0.023	1.00000	0.197531	0.00348	0.00054	1.00000		0.3266	0.3120	0.0410	0.3522	0.68287	0.44212	0.39300	Beverly	6
Andre	2	0.4444	0.059	0.023	1.00000	0.197531	0.00348	0.00054	1.00000		0.3266	0.3120	0.0410	0.3522	0.68287	0.44212	0.39300	Andre	6
Carol	3	0.3333	0.056	0.000	0.81155	0.111111	0.00314	0.00000	0.65861		0.2449	0.2962	0.0000	0.2858	0.76035	0.34007	0.30904	Carol	8
Fernando	4	0.5556	0.071	0.231	1.12913	0.308642	0.00504	0.05358	1.27493		0.4082	0.3755	0.4097	0.3977	0.30204	0.67986	0.69240	Fernando	1
Garth	5	0.5556	0.071	0.231	1.12913	0.308642	0.00504	0.05358	1.27493		0.4082	0.3755	0.4097	0.3977	0.30204	0.67986	0.69240	Garth	1
Ed	6	0.3333	0.056	0.000	0.81155	0.111111	0.00314	0.00000	0.65861		0.2449	0.2962	0.0000	0.2858	0.76035	0.34007	0.30904	Ed	8
Diane	7	0.6667	0.067	0.102	1.36572	0.444444	0.00449	0.01037	1.86519		0.4899	0.3543	0.1803	0.4810	0.50847	0.67111	0.56894	Diane	4
Heather	8	0.3333	0.067	0.389	0.55609	0.111111	0.00449	0.15123	0.30924		0.2449	0.3543	0.6883	0.1959	0.37651	0.75166	0.66626	Heather	3
Ike	9	0.2222	0.048	0.222	0.13649	0.049383	0.00230	0.04938	0.01863		0.1633	0.2539	0.3933	0.0481	0.62922	0.41013	0.39460	Ike	5
Jane	10	0.1111	0.034	0.000	0.03169	0.012346	0.00116	0.00000	0.00100		0.0816	0.1798	0.0000	0.0112	0.94840	0.00000	0.00000	Jane	10
						1.36083	0.18909	0.56500	2.83922	+ Ideal	0.4899	0.3755	0.6883	0.4810					
										- Ideal	0.0816	0.1798	0.0000	0.0112					

Fig. 6: Node ranking based on 4 closeness criteria

	#	X DC	Closeness	Between	EC	AC	Squares and sums					R					S+	S-	C		
Beverly	1	0.4444	0.059	0.023	1.00000	1.00000	0.197531	0.00348	0.00054	1.00000	1.00000	0.3266	0.3120	0.0410	0.3522	0.31622777	0.68287	0.44212	0.39300	Beverly	6
Andre	2	0.4444	0.059	0.023	1.00000	1.00000	0.197531	0.00348	0.00054	1.00000	1.00000	0.3266	0.3120	0.0410	0.3522	0.31622777	0.68287	0.44212	0.39300	Andre	6
Carol	3	0.3333	0.056	0.000	0.81155	1.00000	0.111111	0.00314	0.00000	0.65861	1.00000	0.2449	0.2962	0.0000	0.2858	0.31622777	0.76035	0.34007	0.30904	Carol	8
Fernando	4	0.5556	0.071	0.231	1.12913	1.00000	0.308642	0.00504	0.05358	1.27493	1.00000	0.4082	0.3755	0.4097	0.3977	0.31622777	0.30204	0.67986	0.69240	Fernando	1
Garth	5	0.5556	0.071	0.231	1.12913	1.00000	0.308642	0.00504	0.05358	1.27493	1.00000	0.4082	0.3755	0.4097	0.3977	0.31622777	0.30204	0.67986	0.69240	Garth	1
Ed	6	0.3333	0.056	0.000	0.81155	1.00000	0.111111	0.00314	0.00000	0.65861	1.00000	0.2449	0.2962	0.0000	0.2858	0.31622777	0.76035	0.34007	0.30904	Ed	8
Diane	7	0.6667	0.067	0.102	1.36572	1.00000	0.444444	0.00449	0.01037	1.86519	1.00000	0.4899	0.3543	0.1803	0.4810	0.31622777	0.50847	0.67111	0.56894	Diane	4
Heather	8	0.3333	0.067	0.389	0.55609	1.00000	0.111111	0.00449	0.15123	0.30924	1.00000	0.2449	0.3543	0.6883	0.1959	0.31622777	0.37651	0.75166	0.66626	Heather	3
Ike	9	0.2222	0.048	0.222	0.13649	1.00000	0.049383	0.00230	0.04938	0.01863	1.00000	0.1633	0.2539	0.3933	0.0481	0.31622777	0.62922	0.41013	0.39460	Ike	5
Jane	10	0.1111	0.034	0.000	0.03169	1.00000	0.012346	0.00116	0.00000	0.00100	1.00000	0.0816	0.1798	0.0000	0.0112	0.31622777	0.94840	0.00000	0.00000	Jane	10
							1.36083	0.18909	0.56500	2.83922	3.16228	+ Ideal	0.4899	0.3755	0.6883	0.4810	0.31623				
												- Ideal	0.0816	0.1798	0.0000	0.0112	0.3162				

Fig. 7: Node ranking based on 4 closeness criteria and all equal Asset Critieria–of course there is no difference. This will only happen if the AC values are not all equal.

Node	TOPSIS Rank
4	first
5	first
8	third
7	fourth
9	fifth
1	sixth
2	sixth
3	eighth
6	eighth
10	tenth

TABLE II: Rank

Using Table II, we see the rank of each node (alternate) based on DC, CC, BC, or EC. When we use TOPSIS with equal weighting for each criteria we arrive at the above ranking, which does not agree with any of the other four!

Hence, TOPSIS can be used successfully when we have to balance different measures of centrality and node importance against each other.

VI. User Manual

This program uses SageMath and Python to calculate the TOPSIS of a graph based on several measures. These measures are: Closeness Centrality (CC); Betweenness Centrality (BC); Degree Centrality (DC); and Eigenvector Centrality (EC). It may also take in optional Asset Critically (AC) measurements. AC takes in a separate weights file than do the centrality measurements. For the AC weights file, use a plain editor and save as a CSV file. The first number represents the node number and should be an integer. There should be a comma to separate the first number and the second number. The second number represents the weight and should be a floating point decimal. For example 1 would be 1.0 and .1 would be 0.1.

The script takes in an input file of JSON for the specifications of the graph. Alternatively, it also accepts a TSV file defining the edges of the graph. In both cases, nodes are defined as integers greater than zero. In the following steps, replace any words inside the brackets ([. . .]) with your (the user's) information. This program is meant to be run on a linux machine.

1) To install in SageMath Python environment, go to command line and type: `sage -python -m pip install pathlib argparse`
2) To print a short description of all command line options for the script, type: `sage -python[program name].py -h`
3) The standard line to run the script using a JSON file is as follows:
   `sage -python [program name].py [file name].js -[optional measure] [optional weight]`
4) To run the script using a TSV file, replace .js with .tsv
5) To add an optional argument for asset criticality (AC), type: `sage -python [program name].py [file name].js -[optional measure] [optional weight] -AC [file name].csv`
6) The output of your program should create a text file which contains data similar to this:

Rankings for graph kite.js

Rank	Number	TOPSIS Closeness C Value
1	8	1.000000
2	4	0.595238
2	5	0.595238
4	9	0.571429
5	7	0.261905
6	1	0.059524
6	2	0.059524
8	3	0.000000
8	6	0.000000
8	10	0.000000

TABLE 3: Rankings for BC

`sage -python topsisRK.py kite.js -BC 1.0`

VII. Experiment

Once again looking at the Kite graph from Figure 3, we combine the four different centrality measures into combinations of pairs. First, we initialize one of the measures to 0.1 while keeping the other at 1.0. Then we shuffle which measurements are paired with each other and repeat these steps for each pair.

Below are a series of tables that show a few results of the different centrality measurement pairs and their respective weights. Each table contains the node ranking, node number, and centrality values.

Rank	Node Number	TOPSIS Closeness C Value
1	8	0.965644
2	4	0.595898
2	5	0.595898
4	9	0.569989
5	7	0.266768
6	1	0.068656
6	2	0.068656
8	3	0.023161
8	6	0.023161
10	10	0.000000

TABLE 4: Rankings of BC = 1.0, DC = 0.1

```
sage -python topsisRK.py kite.js -BC 1.0 -DC 0.1
```

Rank	Node Number	TOPSIS Closeness C Value
1	7	0.889427
2	4	0.792331
2	5	0.792331
4	1	0.582399
4	2	0.582399
6	8	0.419774
7	3	0.390916
7	6	0.390916
9	9	0.216527
10	10	0.000000

TABLE 5: Rankings of BC = 0.1, DC = 1.0

```
sage -python topsisRK.py kite.js -BC 0.1 -DC 1.0
```

Rank	Node Number	TOPSIS Closeness C Value
1	8	0.960232
2	4	0.596197
2	5	0.596197
4	9	0.568828
5	7	0.268307
6	1	0.076071
6	2	0.076071
8	3	0.038360
8	6	0.038360
10	10	0.000000

TABLE 6: Rankings of BC = 1.0, EC = 0.1

```
sage -python topsisRK.py kite.js -BC 1.0 -EC 0.1
```

Rank	Node Number	TOPSIS Closeness C Value
1	7	0.902491
2	4	0.815627
2	5	0.815627
4	1	0.702901
4	2	0.702901
6	3	0.570287
6	6	0.570287
8	8	0.408707
9	9	0.110554
10	10	0.000000

TABLE 7: Rankings of BC = 0.1, EC = 1.0

```
sage -python topsisRK.py kite.js -BC 0.1 -EC 1.0
```

According to Tables 4 & 5, we observe that ranking varies depending on which centrality measures we use. When BC has a greater weight than DC, node 8 is ranked first. On the other hand, if DC has a greater weight, then node 7 is ranked much higher than node 8.

From these results we see that the ranking of a node is not only affected by its centrality but also by its position in the network structure. Node 8 would have the highest BC in the Kite graph because it is placed in a position connected to the greatest number of nodes that are not connected to each other. Conversely, node 7 would be ranked much higher in than node 8 in terms of DC because it has more alternative routes than other nodes to reach its goal.

Tables 5 & 7 show that node 7 consistently has a high ranking for both DC and EC. As stated before, DC is the number of links connected to a node and the basic concept behind EC is that an important node is connected to important neighbors. Given the position of node 7 in the Kite graph, it would make sense for it to be ranked first for both centrality measures. Note that even though EC and DC rank the nodes the same, the TOPSIS Closeness C values for them are normalized differently (think of 100 being higher than 15, and 20 being higher than 15).

Further testing is done by using asset criticality (AC). We start by taking the different centrality measures and finding the node rank. Then we take the highest ranking node and change its asset criticality to 0.1. The lowest ranking node is given a higher asset criticality of 3.0. The other nodes are kept constant with an asset criticality of 1.0. Below are a series of tables that show the rankings of each centrality measure after asset criticality has been changed between high and low ranked nodes.

Rank	Node Number	AC Weights = 2nd column weightsRK.csv
1	10	3.0
2	8	0.1
3	4	1.0
3	5	1.0
5	9	1.0
6	7	1.0
7	1	1.0
7	2	1.0
9	3	1.0
9	6	1.0

TABLE 8: Switched AC weights for BC

```
sage -python topsisRK.py kite.js -BC 1.0 -AC weightsRK.csv
```

Rank	Node Number	AC Weights = 2nd column weightsRK.csv
1	10	3.0
2	4	1.0
2	5	1.0
4	1	1.0
4	2	1.0
6	7	0.1
7	3	1.0
7	6	1.0
7	8	1.0
10	9	1.0

TABLE 9: Switched AC weights for DC

```
sage -python topsisRK.py kite.js -DC 1.0 -AC weightsRK.csv
```

Rank	Node Number	AC Weights = 2nd column weightsRK.csv
1	10	3.0
2	5	1.0
3	7	1.0
3	8	1.0
5	1	1.0
5	2	1.0
7	3	1.0
7	6	1.0
9	9	1.0
10	4	0.1

TABLE 10: Switched AC weights for CC

```
sage -python topsisRK.py kite.js -CC 1.0 -AC weightsRK.csv
```

Rank	Node Number	AC Weights = 2nd column weightsRK.csv
1	10	3.0
2	4	1.0
2	5	1.0
4	1	1.0
4	2	1.0
6	3	1.0
6	6	1.0
8	7	0.1
9	8	1.0
10	9	1.0

TABLE 11: Switched AC weights for EC

```
sage -python topsisRK.py kite.js -EC 1.0 -AC weightsRK.csv
```

Looking at Table 11, we see the nodes behave as expected. When we change asset criticality, the lowest ranking node for each centrality measure becomes the highest ranking node. The nodes which were ranked between the highest and lowest nodes stayed in the middle, and the previous first ranked node gets ranked much lower.

VII-A. Sagemath Python Code: topsisRK.py

```
#!/usr/bin/env sage

# Author : Jonathan Decker
#
# Description : Script that uses SageMath to calculate the TOPSIS
# of a graph based on several measures. The script takes an input file
# of JSON format for the specification of the graph. This is identical
# to the python dictionary format used in sage. Alternatively it excepts
# a TSV file defining the edges of the graph. In both cases nodes are
# defined as integers greater than zero.
#
# usage:   topsisRK.py [-h] [-CC weight] [-DC weight] [-EC weight] [-BC weight]
#                [-AC weights]
#                graph
#
# Package Dependences : pathlib, argparse, sage
#
# To install in sage math python environment
# sage -python -m pip install pathlib argparse

import re
import sys
import json
import math
import string
import argparse
from pathlib import Path
from sage.all import Graph,vector,Rational,IntegerRange,AlgebraicNumber
import numpy as np
from scipy.stats import rankdata

keyRE        = re.compile("(\d+)\s*:")
intRE        = re.compile("^\d+$")
floatRE      = re.compile("^\d+\.\d+$")
floatRangeRE = re.compile("^\d+\.\d+\?$")
commentRE    = re.compile("^\%")

# declaration of the supported measures
possibleMeasures = { 'DC' : ['Degree Centrality',      lambda : G.centrality_degree()],
                     'CC' : ['Closeness Centrality',   lambda : G.centrality_closeness()],
                     'BC' : ['Betweenness Centrality', lambda : G.centrality_betweenness()],
                     'EC' : ['Eigenvector Centrality', lambda : centrality_eigenvector(G) ] }

measureWeights = {}

measureList = ','.join(possibleMeasures.keys())
measureHelp = ', '.join(['%s (%s)' % (v[0],k) for k,v in possibleMeasures.iteritems()])

parser = argparse.ArgumentParser(description='Script that uses Sage Math to calculate the TOPSIS of a graph based on several measures.  Supported measures include: %s' % measureHelp)
parser.add_argument('file', metavar='graph', type=str, help='graph file (JSON | TSV)')

for k,v in possibleMeasures.iteritems():
    parser.add_argument('-' + k,  metavar='weight', type=float, help='calculate ' + v[0] + ' measure weighted by the value.')

parser.add_argument('-AC',  metavar='weights', type=str, help='calculate AC measure based on weights in CSV file')

args = parser.parse_args()

def cast_int(d):
    ret = {}
    for k,v in d.iteritems():
        if (isinstance(k,str) or isinstance(k,unicode)) and intRE.match(k):
            ret[int(k)] = v
        else:
            ret[k] = v
    return ret

# computes a centrality measurement of the nodes in graph (g) with eigenvectors
def centrality_eigenvector(g):
    print "    calculating eigenvectors..."
    eigenVectors = g.eigenvectors()
```

```
    print "    calculating centrality..."
    eigenVectors = sorted( eigenVectors, key=lambda col: col[0], reverse=True )
    v = vector(eigenVectors[0][1][0])
    e = v.normalized()
    #eigen = {j+1: float(k) for j,k in enumerate(e)}

    eigen = {}
    for j,k in enumerate(e):
        if isinstance(k,Rational):
            fK = float(k)
        elif isinstance(k,AlgebraicNumber):
            sK = str(k)
            if floatRangeRE.match(sK):
                sK = sK[:-1]
                fK = float(sK)
        else:
            fK = float(k)
        eigen[j+1] = fK
    return eigen

def buildDecisionMatrix( m, n, measures ):
    D = np.zeros((m,n))
    nIdx = 0
    for name,pairs in measures.iteritems():
        for mIdx,value in pairs.iteritems():
            D[mIdx-1][nIdx] = value
        nIdx += 1
    return D

def buildWeightVector( measures, weights ):
    W = np.zeros(len(weights.keys()))
    idx = 0
    for name,pairs in measures.iteritems():
        print name
        W[idx] = weights[name]
        idx += 1
    return W

def handleJSON( inputJSON ):
    graphDic = {}
    with inputJSON.open() as f:
        s = f.read()
        s = keyRE.sub(r'"\g<1>":', s)
        graphDic = json.loads(s)
        graphDic = cast_int(graphDic)
    return graphDic

def handleTSV( inputTSV ):
    graphDic = {}
    with inputTSV.open() as f:
        header = f.readline()
        l = f.readline()
        while l:
            if not commentRE.match(l):
                a,b = [int(x) for x in l.split()]
                if not graphDic.has_key(a):
                    graphDic[a] = []
                graphDic[a].append(b)
            l = f.readline()
    return graphDic

def handleFile( fileName ):
    inputFile = Path(fileName)
    if inputFile.exists() and not inputFile.is_file():
        print "ERROR: %s is not a file" % inputFile
        sys.exit(-1)

    if not inputFile.exists():
        print "ERROR: Cannot find file %s" % inputFile
        sys.exit(-1)

    graphDic = {}
```

```
144
145     if inputFile.suffix == ".tsv":
146         graphDic = handleTSV(inputFile)
147     elif inputFile.suffix == ".js":
148         graphDic = handleJSON(inputFile)
149     else:
150         print "ERROR: File %s not a supported format" % inputFile
151         sys.exit(-1)
152
153     return graphDic
154
155 def handleWeightsCSV( fileName ):
156     inputFile = Path(fileName)
157     if inputFile.exists() and not inputFile.is_file():
158         print "ERROR: %s is not a file" % inputFile
159         sys.exit(-1)
160     if not inputFile.exists():
161         print "ERROR: Cannot find file %s" % inputFile
162         sys.exit(-1)
163     if inputFile.suffix != ".csv":
164         print "ERROR: Weights file %s not a supported format (csv)" % inputFile
165
166     weights = {}
167     with inputFile.open() as f:
168         l = f.readline()
169         while l:
170             items = l.split(',')
171             if len(items) == 2 and intRE.match(items[0]) and floatRE.match(items[1]):
172                 weights[int(items[0])] = float(items[1])
173             l = f.readline()
174
175     return weights
176
177 def getWeights( weightFile, normalize ):
178     weights = handleWeightsCSV(args.AC)
179     measuresAC = {}
180     for node in nodeNames:
181         if not weights.has_key(node):
182             print "Warning: no weight for %d default to 1.0" % node
183             measuresAC[node] = 1.0
184         else:
185             measuresAC[node] = weights[node]
186
187     if normalize:
188         maxVal = max(measuresAC.values())
189         for idx in measuresAC.keys():
190             measuresAC[idx] /= maxVal
191
192     return measuresAC
193
194 graphDic = handleFile(args.file)
195 measureNames = []
196
197 argsDict = vars(args)
198
199 for k in possibleMeasures.keys():
200     if argsDict[k] is not None:
201         measureWeights[k] = argsDict[k]
202         measureNames.append(k)
203
204 #measureNames = args.measures
205
206 if len(measureNames) != 0 and len(graphDic) != 0:
207     G = Graph(graphDic)
208
209     numNodes = G.order()
210     nodeNames = G.vertices()
211
212     print "Calculating Measures..."
213
214     # measures being used in TOPSIS
215     measures = {}
```

```
    # set up AC weights from file
    if args.AC is not None:
        measures["AC"] = getWeights(args.AC,False)
        measureWeights["AC"] = 1.0

    for name in measureNames:
        if possibleMeasures.has_key(name):
            print "Calculating %s..." % possibleMeasures[name][0]
            measures[name] = possibleMeasures[name][1]()
        else:
            print "Warning: %s is not a supported measure" % name
            if name.find(',') >= 0:
                print "    NOTE: list of measures separated by spaces, not commas"

    numMeasures = len(measures)

    if numMeasures < 1:
        print "ERROR: No supported measures requested"
        sys.exit(-1)

    #start of TOPSIS calculation

    print "Calculating TOPSIS..."

    D = buildDecisionMatrix(numNodes,numMeasures,measures)

    W = buildWeightVector( measures, measureWeights )

    # Equation 6
    norms = np.linalg.norm(D,axis=0)
    R = np.divide(D,norms)

    # Equation 7
    R = np.multiply(R,W) # weight metrics

    # Equations 8-13
    posIdeal = np.amax(R,axis=0)
    negIdeal = np.amin(R,axis=0)

    posSeperation = np.linalg.norm(np.subtract(R,posIdeal),axis=1)
    negSeperation = np.linalg.norm(np.subtract(R,negIdeal),axis=1)

    # Equation 14
    C = np.divide(negSeperation,np.add(posSeperation,negSeperation))

    ranks = rankdata(np.subtract(np.ones(numNodes),C),'min')

    rankedValues = zip(ranks,range(1,numNodes+1),C)

    rankedValues = sorted(rankedValues, key=lambda x: x[0])

    # Printing the rankings
    print "Rankings for graph %s\n" % args.file
    print "Rank |   # | closeness"
    print "----------------------"
    for i,c in enumerate(rankedValues):
        print " %3d | %3d | %.6f" % c

    # Draw graph
    inputFile = Path(args.file)
    p = G.plot()
    p.save(inputFile.stem + '.png')
else:
    print "ERROR: No measures specified or graph is empty"
```

REFERENCES

[1] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web*, pages 519–528. ACM, 2012.

[2] Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[3] Chen Chen, Hanghang Tong, B. Aditya Prakash, Charalampos Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. Node immunization on large graphs: Theory and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(11):1–13, November 2014.

[4] Carlos D Correa, Tarik Crnovrsanin, and Kwan-Liu Ma. Visual reasoning about social networks using centrality sensitivity. *Visualization and Computer Graphics, IEEE Transactions on*, 18(1):106–120, 2012.

[5] The Sage Developers. Sagemath, the Sage Mathematics Software System (Version 7.2). `http://www.sagemath.org`.

[6] Yuxian Du, Cai Gao, Yong Hu, Sankaran Mahadevan, and Yong Deng. A new method of identifying influential nodes in complex networks based on TOPSIS. *Physica A*, 399:57–69, 2014.

[7] Yuxian Dua, Cai Gaoa, Yong Hub, Sankaran Mahadevanc, and Yong Deng. A new method of identifying influential nodes in complex networks based on topsis. *Physica A*, 399:57–69, 2014.

[8] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1979.

[9] William Goffman and Vaun A. Newill. Generalization of epidemic theory: An application to the transmission of ideas. *Nature*, 204:225–228, October 1964.

[10] P.R. Gould. On the geographical interpretation of eigenvalues. *Transactions of the Institute of British Geographers*, (42):53–86, December 1967.

[11] Adrien Guille, Hakim Hacid, Cécile Favre, and Djamel A Zighed. Information diffusion in online social networks: A survey. *ACM SIGMOD Record*, 42(2):17–28, 2013.

[12] Jiantao Hua, Yuxian Dua, Hongming Mob, Daijun Weic, and Yong Deng. A modified weighted topsis to identify influential nodes in complex networks. *Physica A*, 444:73–85, 2016.

[13] C.L. Hwang and K. Yoon. *Multiple Attribute Decision Making: Methods and Applications*. Springer-Verlag, New York, 1981.

[14] Aarti Jackson and B.V.R. Reddy. Node centrality in wireless sensor networks: Importance, applications and advances. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, pages 127–131, 2013.

[15] Matthew O. Jackson. *Social and Economic Networks*. Princeton University Press, 2008.

[16] Aarti Jain and B.V.R. Reddy. Eigenvector centrality based cluster size control in randomly deployed wireless sensor networks. *Expert Systems with Applications: An International Journal*, 42(5):2657–2669, April 2015.

[17] Anya Kim and Myong H. Kang. Determining asset criticality for cyber defense. Memorandum Report NRL/MR/5540–1-9350, Naval Research Laboratory, September 2011.

[18] Anya Kim, Myong H. Kang, Jim Z. Lou, and Alex Velazquez. A framework for event prioritization in cyber network defense. Memorandum Report NRL/MR/554—14-9541, Naval Research Laboratory, September 2014.

[19] Peter Lancaster and Miron Tismenetsky. *The Theory of Matrices*. Academic Press, 2 edition, 1985.

[20] Alireza Louni and K.P. Subbalakshmi. Diffusion of information in social networks. In *Social Networking*, pages 1–22. Springer, 2014.

[21] Todd E Malinick, David B Tindall, and Mario Diani. Network centrality and social movement media coverage: A two-mode network analytic approach. *Social Networks*, 35(2):148–158, 2013.

[22] Brian McRoberts. Internet plays integral role in decision-making: Study. *China Daily*, 13(11), September 2010.

[23] Piet Van Mieghem, Dragan Stevanovic , Fernando Kuipers, Cong Li, and Ruud van de Bovenkamp. Decreasing the spectral radius of a graph by link removals. *Physical Review E*, 84(1):016101(12), 2011.

[24] Ira S. Moskowitz, Paul Hyden, and Stephen Russell. Network topology and mean infection times. *Social Network Analysis and Mining*, 6(34), June 2016.

[25] Akira Namatame and Shu-Heng Chen. *Agent-Based Modeling and Network Dynamics*. Oxford University Press, 1st edition, 2016.

[26] M.E.J. Newman. *Networks an Introduction*. Oxford, 2010.

[27] Serafim Opricovic and Gwo-Hshiung Tzeng. Compromise solution by MCDM methods. *European Journal of Operational Research*, 156:445–455, 2004.

[28] Juan G. Restrepo, Edward Ott, and Brian R. Hunt. Characterizing the dynamical importance of network nodes and links. *Physical Review Letters*, 97(9):94102(4), September 2006.

[29] John Scott. *Social network analysis*. Sage, 2012.

[30] E. Triantaphyllou. *Multi-Criteria Decision Making Methods: A Comparative Study*. Kluwer Academic, Dordrecht, 2000.