Noname manuscript No.
(will be inserted by the editor)

Centrality Measures for Node-Weighted Networks via Line Graphs and the Matrix Exponential

Omar De la Cruz Cabrera · Mona Matar · Lothar Reichel

the date of receipt and acceptance should be inserted later

Abstract This paper is concerned with the identification of important nodes in node-weighted graphs by applying matrix functions, in particular the matrix exponential. Many tools that use an adjacency matrix for a graph have been developed to study the importance of the nodes in unweighted or edge-weighted networks. However, adjacency matrices for node-weighted graphs have not received much attention. The present paper proposes to use a line graph associated with a node-weighted graph to construct an edge-weighted graph, that can be analyzed with available methods. Both undirected and directed graphs with positive node weights are considered. We show that when the weight of a node increases, the importance of this node in the graph increases as well, provided that the adjacency matrix is suitably scaled. Applications to real-life problems are presented.

Key words: network analysis, node weight, node importance, line graph, matrix exponential

AMS subject classifications: 05C50, 15A16, 65F60, 90B10

1 Introduction

A network is a set of entities, commonly referred to as *vertices* or *nodes*, that are connected by *edges*. Mathematically, networks can be represented by *graphs*. The mathematical and computational analysis of a graph can give valuable information about the network it models, even though most of the particular attributes of the

Omar De la Cruz Cabrera
Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA. E-mail: odelacru@kent.edu

Mona Matar
Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA. E-mail: mmatar2@kent.edu

Lothar Reichel
Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA. E-mail: reichel@math.kent.edu

individual vertices and edges are ignored. A few examples of the many phenomena that can be modeled by graphs are: interactions among people or animals in a group (social networks), gene regulatory interactions, telecommunication, power and transportation infrastructure, and scientific collaboration; see, e.g., [2,5,16,29,32,35,41, 43] for these and other applications.

A major concern in network analysis is determining which nodes are "important" in some sense. The importance of a node depends not only on how many neighboring nodes it has, but also on the importance of the neighboring nodes. For instance, consider a graph in which the nodes represent scientific papers and the edges represent citations. A paper is important if it is cited by many important papers; at the same time, an important paper conveys importance to papers that it cites and, to a lesser extent, papers that cite an important paper also may be of interest. Network analysis can help determine which nodes (papers) contribute the most to broadcasting or receiving information through the network.

A popular approach to determine the importance of a node in a graph is to use the exponential of the adjacency matrix A for the graph and compute the subgraph centrality of the nodes. The subgraph centrality of node i is defined as the ith diagonal entry of $\exp(A)$. Node i is considered important if $[\exp(A)]_{ii} \geq [\exp(A)]_{jj}$ for all j; see Estrada and Hatano [17] for a discussion on this importance measure for undirected graphs. The related importance measures $[\exp(A)\mathbf{1}]_i$ and $[\exp(A^T)\mathbf{1}]_i$ for node i as a broadcaster or receiver of information, respectively, for directed graphs, where $\mathbf{1}$ denotes the vector with all entries one, are discussed, e.g., in [7,12]. Here the importance of node i as a broadcaster increases with the relative size of $[\exp(A)\mathbf{1}]_i$. Thus, node i is a most important broadcaster if

$$[\exp(A)\mathbf{1}]_i \geq [\exp(A)\mathbf{1}]_j \quad \forall j. \tag{1.1}$$

Analogously, node i is a most important receiver of information if

$$[\exp(A^T)\mathbf{1}]_i \geq [\exp(A^T)\mathbf{1}]_j \quad \forall j. \tag{1.2}$$

The subgraph centrality as well as the importance measures (1.1) and (1.2) take into account both the number of nodes that node i is connected to and the importance of these nodes. Further discussions on these measures and additional references are provided in Sections 2 and 3. For additional discussions on importance measures, we refer to [16,18,20,25,32]. These notions of importance are not equivalent, in general, and tend to capture different aspects of the idea of importance.

Measures of node importance are sometimes referred to as *centrality* measures. In this manuscript, since we consider directed networks, we prefer to use the term *importance*. For example, in a directed network with a universal source and a universal sink, those two nodes would fit the informal notion of "important," but not so much the informal notion of "central".

It is often meaningful to assign weights to edges or nodes. For example, if each node represents a city and each edge represents a road, an edge weight may represent the capacity of transportation of the road. Edge-weighted networks have received considerable attention in the literature; see, e.g., [4,10,31,34,44]. It can also be useful to assign weights to nodes. The interpretation of node weights depends on the context of the model. For instance, in a network that models a part of the brain, where each node corresponds to a region of the brain, node weights may be chosen proportional to the size of the region of interest [1]. In networks, in which each node corresponds to a city and the edges are roads between cities, a node weight may be chosen proportional to the number of restaurants in a city [27]. Node weights also may measure precipitation in a geographical region in a climate network [40]. We will in Subsection 7.1 analyze genotype mutations with the aid of node-weighted networks. However, despite many applications of node-weighted networks, the construction of suitable adjacency matrices for their graphs has, to the best of our knowledge, not received much attention in the literature.

This paper is concerned with the identification of the most important nodes of a node-weighted network by using matrix functions, in particular the matrix exponential. A main challenge is the construction of a suitable adjacency matrix. Our approach is to transform a given node-weighted graph to an edge-weighted graph by applying a line graph associated with the given graph. We describe several ways to construct line graphs, and apply the exponential to adjacency matrices associated with the line graph to determine the most important nodes in the line graph by using analogues of the formulas (1.1) and (1.2) with the adjacency matrix A for the graph replaced by an adjacency matrix E for the line graph. The entries of the analogues of the vectors (1.1) or (1.2) so obtained provide edge weights that take into account not only to how many edges an edge is connected to, but also to their importance. These edge weights define an edge-weighted adjacency matrix $\widetilde{A}$ for the original graph. The importance of the nodes of the original graph is computed by formulas analogous to (1.1) and (1.2) with the A replaced by $\widetilde{A}$. This approach allows us to incorporate node weights for the original graphs. Both undirected and directed graphs are considered. We also discuss how increasing an edge weight affects the importance of the node.

The organization of this paper is as follows. Section 2 introduces graphs and associated adjacency matrices. The use of the matrix exponential function for edge-weighted graphs is reviewed in Section 3. We discuss ways to transform a node-weighted graph into an edge-weighted graph in Section 4, and Section 5 shows results on how node importance (according to our definitions) changes when the weight of an edge is modified. Section 6 is concerned with the identification of the most important node(s) and edge(s) of a node-weighted graph, and Section 7 presents real-life applications of our methods. Computed illustrations are provided in most sections. A discussion on numerical approaches for large networks is provided in Section 8, and concluding remarks can be found in Section 9.

2 Graphs

This section reviews definitions and well-known properties of graphs and the associated adjacency matrices. For more thorough discussions; see, e.g., [16,32].

2.1 Notation

Mathematically, a network is represented by a *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consisting of a set $\mathcal{V} = \{v_i\}_{i=1}^n$ of *vertices* or *nodes*, and a set $\mathcal{E} = \{e_k\}_{k=1}^m$ of *edges*, which are the links between the nodes. A graph is said to be *directed* if at least one of the edges has a direction; otherwise the graph is *undirected.* For directed graphs, each element of $\mathcal{E}$ corresponds to an ordered pair $e_k = (v_i, v_j)$ of elements of $\mathcal{V}$, and we say that e_k

incides on v_j, *exsurges* from v_i, and *connects* v_i and v_j. In the undirected case, each element of $\mathcal{E}$ corresponds to an unordered pair $e_k = \{v_i, v_j\}$ of elements of $\mathcal{V}$, and we say that e_k *incides* on both v_i and v_j, and *connects* v_i and v_j (and also v_j and v_i). In some cases we will require that "self-loops", the case when e_k connects v_i to itself, do not exist. We assume that there are no multiple edges between any pair of vertices. The *underlying undirected graph* of a directed graph has the same set of vertices, with all directed edges replaced by undirected ones (eliminating duplicate edges, if any).

Whether directed or not, a network is said to be *edge-weighted* (or *node-weighted*), if there is a number assigned to each edge (or node, respectively). We refer to these numbers as "weights." Graphs may be both edge-weighted and node-weighted, but this case is beyond the scope of the present paper. The interpretation of the weights depends on the application. In general, node weights correspond to the "size" of a node, while edge weights indicate a capacity or speed of transportation, or the reciprocal of a transfer or communication cost. In this paper we only consider positive weights, although negative weights may be meaningful in some contexts.

For undirected unweighted graphs, the *degree* of a node is defined as the number of edges inciding on it; for directed unweighted graphs, we identify the *indegree* of a node as the number edges inciding on at it, and the *outdegree* of a node as the number of edges exsurging from it.

This paper is concerned with node-weighted graphs, for which each node is assigned a positive weight. All edges have weight one. When constructing an associated edge-weighted graph, its nodes will have weight one, and its edges will have weights as described in Section 4.

2.2 Matrix Representation of Graphs

Algebraic Graph Theory is a powerful approach for the analysis of graphs. It is based on Linear Algebra, and uses matrices to represent graphs; see, e.g., [14, 16, 21, 32] for more details. This section reviews a few concepts needed for this paper.

2.2.1 Adjacency and Incidence Matrices of Unweighted Graphs

For an unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V} = \{v_i\}_{i=1}^n$ and edge set $\mathcal{E} = \{e_k\}_{k=1}^m$, and without multiple edges, the *adjacency matrix* of $\mathcal{G}$ is an $n \times n$ matrix $A = [A_{ij}]$ with $A_{ij} = 1$ if there exists an edge that points from node v_i to node v_j, and $A_{ij} = 0$ otherwise. The matrix A is symmetric if the graph is undirected; for directed graphs, A is generally nonsymmetric. If a directed unweighted graph has a symmetric adjacency matrix, then the directed graph can be replaced by its undirected underlying graph.

While the adjacency matrix of a graph gives a representation of node-to-node connections, the *incidence matrix* is determined by node-to-edge connections. If the network is undirected, the incidence matrix of $\mathcal{G}$ is an $n \times m$ matrix $B = [B_{ik}]$ with $B_{ik} = 1$ if e_k incides on v_i, and $B_{ik} = 0$ otherwise. Each column of B has exactly two nonzero entries, which are equal to one, unless the corresponding edge is a self-loop, in which case exactly one entry is one. If there are no self-loops, then $BB^T = A + D$, where $D = [D_{ij}]$ is a diagonal matrix with the diagonal entry D_{ii} equal to the degree of v_i. Throughout this paper the superscript ^{T} denotes transposition.

For directed networks, there are various ways to define an incidence matrix; see, e.g., [9,21]. In this paper we will adopt the definition used in [13]. Thus, for a directed and unweighted graph $\mathcal{G}$, the *incidence matrix* and *exsurgence matrix* of $\mathcal{G}$ are the $n \times m$ matrices $B^{\mathrm{i}} = [B^{\mathrm{i}}_{ik}]$ and $B^{\mathrm{e}} = [B^{\mathrm{e}}_{ik}]$, respectively, with $B^{\mathrm{i}}_{ik} = 1$ if e_k incides on v_i, $B^{\mathrm{e}}_{ik} = 1$ if e_k exsurges from v_i, and all other entries vanish. We say that the *head* of an edge e_k is at v_i if e_k incides on v_i, and that the *tail* of e_k is at v_j if e_k exsurges from v_j. Informally, an edge e_k is said to *transmit information* from node v_j to node v_i, if the head of e_k is at v_i and its tail is at v_j. Each column of B^{i} and B^{e} has exactly one nonzero entry, equal to unity. We have that

$$A = B^{\mathrm{e}} B^{\mathrm{i}T}. \tag{2.1}$$

2.2.2 Adjacency and Incidence Matrices of Edge-Weighted Graphs

Let the edges of the graph $\mathcal{G}$ have positive weights and denote the associated weighted adjacency matrix by $\widetilde{A}$. Thus, the $(ij)^{\mathrm{th}}$ entry of $\widetilde{A}$ is the weight of the edge from node v_i to node v_j. We refer to the adjacency matrix $\widetilde{A}$ as *edge-weighted.* The "unweighted" adjacency matrix A that is associated with $\widetilde{A}$ has all edge weights equal to one. Thus, the entries of A belong to $\{0, 1\}$.

Consider the unweighted adjacency matrix (2.1) associated with the edge-weighted graph $\mathcal{G}$, and let the matrix $Z = \mathrm{diag}(z_1, z_2, \ldots, z_m)$ hold the edge weights $z_1, z_2, \ldots, z_m$ of the graph. Then the weighted adjacency matrix for the graph $\mathcal{G}$ can be written as

$$\widetilde{A} = B^{\mathrm{e}} Z B^{\mathrm{i}T}. \tag{2.2}$$

2.3 Line Graphs

2.3.1 Line Graph of an Undirected Graph

The *line graph* of an undirected unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph $\mathcal{G}^* = (\mathcal{E}, \mathcal{F})$, in which an edge $f \in \mathcal{F}$ connects the nodes $e, e' \in \mathcal{E}$ if and only if there is a node $v \in \mathcal{V}$ such that both e and e' incide on v in $\mathcal{G}$. Line graphs are used to analyze networks in various contexts; see, e.g., [15,22,37,38].

It is easy to show that if B is the incidence matrix of $\mathcal{G}$, then $E = B^T B - 2I_m$ is the adjacency matrix of $\mathcal{G}^*$. Here and below, I_m stands for the identity matrix of order m. We will refer to E as the *line graph adjacency matrix.*

2.3.2 Line Graphs of a Directed Graph

While there is only one natural notion of line graph for undirected graphs, several different line graphs can be associated with a directed unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; see [13]. We will use the line graph $\mathcal{G}^* = (\mathcal{E}, \mathcal{F})$ described by Thulasiraman and Swamy [42], where each edge in $\mathcal{G}$ corresponds to a node in $\mathcal{G}^*$, and an edge exists from node e_i to node e_j in $\mathcal{G}^*$ only if edge e_j in $\mathcal{G}$ emerges from the node that edge e_i points to. The *line graph adjacency matrix* of $\mathcal{G}$ is $E^{\rightarrow} = B^{\mathrm{i}T} B^{\mathrm{e}}$; this is shown in [13].

3 Identifying the Most Important Edges and Nodes in Edge-Weighted Graphs

3.1 Importance of Nodes For Undirected and Directed Graphs

Many methods have been proposed to assess the importance or centrality of nodes in graph. We will focus on methods that use matrix functions, in particular the matrix exponential; see, e.g., [11,12,18,19] for discussions on this approach. Let $\widetilde{A} \in \mathbb{R}^{n\times n}$ be the adjacency matrix of an edge-weighted graph, and let p be a positive integer. Then the matrix entry $[\widetilde{A}^p]_{ij}$ is a sum with one term for each walk of length p starting at v_i and ending at v_j (following edge directions, if the graph is directed), and each term being the product of the weights of the edges in the corresponding walk. Hence, the $(ij)^{\text{th}}$ entry of the matrix function f evaluated at $\widetilde{A}$,

$$f(\widetilde{A}) = \sum_{p=0}^{\infty} c_p \widetilde{A}^p, \tag{3.1}$$

is a weighted sum with terms corresponding to walks of various lengths from node v_i to node v_j. The coefficients c_p are chosen to penalize walks that traverse many edges, because such walks are typically considered less important than walks that traverse few edges. The coefficients generally are chosen to be nonnegative and to eventually decrease as functions of p. A common choice is $c_p = 1/p!$, in which case $f(\widetilde{A}) = \exp(\widetilde{A})$; see, e.g., [18] for a discussion on this matrix function. The term $c_0 I_n$ in (3.1) does not affect the ordering of the diagonal entries and, therefore, is of no importance.

A popular centrality measure for node v_i of a network is the *subgraph centrality*, which is given by $[f(\widetilde{A})]_{ii}$; see, e.g., [11,12,17–19]. Another commonly used measure is

$$[f(\widetilde{A})\mathbf{1}]_i, \quad i = 1,2,\dots,n, \qquad \mathbf{1} = [1,1,\dots,1]^T; \tag{3.2}$$

see, e.g., [7,13,16,18,25]. For undirected graphs, $\widetilde{A}$ is symmetric, and a relatively large value of $[f(\widetilde{A})]_{ii}$ or $[f(\widetilde{A})\mathbf{1}]_i$ indicates that node v_i is important. For directed graphs, the relative size of $[f(\widetilde{A})\mathbf{1}]_i$ shows the importance of node v_i as a broadcaster, and the relative size of $[f(\widetilde{A}^T)\mathbf{1}]_i$ shows its importance as a receiver; see [12] for a thorough discussion of the case when f is the exponential function. In this paper we will use the measures *aggregate downstream reachability*,

$$\text{ADR} = \exp(\widetilde{A})\mathbf{1}, \tag{3.3}$$

and *aggregate upstream reachability*,

$$\text{AUR} = \exp(\widetilde{A}^T)\mathbf{1}. \tag{3.4}$$

The ordering of the ADR entries provides a ranking of all nodes in the network in their role as broadcasters, and the ordering of the AUR entries determines a ranking of the nodes in their role as receivers. The largest values correspond to the most important nodes. For undirected graphs, the ADR- and AUR-values are the same.

3.2 Importance of Edges in Undirected Networks

Let $E \in \mathbb{R}^{m\times m}$ be the line graph adjacency matrix associated with an undirected graph $\mathcal{G}$. The edges of $\mathcal{G}$ may be ranked by comparing the *edge line graph centralities*

$$[\exp(E)]_{kk} = \sum_{p=0}^{\infty} \frac{1}{p!}[E^p]_{kk}, \qquad k = 1, 2, \ldots, m. \tag{3.5}$$

The largest entries are associated with the most important edges; see [13]. Alternatively, one may compare the relative size of the quantities

$${}^{e}\mathrm{LC}_k = [\exp(E)\mathbf{1}]_k, \tag{3.6}$$

which are analogous to (3.2). Again, the largest quantities are associated with the most important edges. These centrality measures also can be applied to line graphs with weighted edges. We will refer to the adjacency matrix associated with a weighted line graph as $\widetilde{E}$.

3.3 Importance of Edges in Directed Graphs

The line graph adjacency matrix $E^{\to}$ of a directed unweighted graph has an entry 1 in position (k, j) if edge e_k passes information to edge e_j through a node, i.e., if the head of edge e_k coincides with the tail of edge e_j. The entries of the matrix $(E^{\to})^2$ tell us whether information is passed from an edge to another edge through two nodes. In other words, $[(E^{\to})^2]_{kj} = 1$ if there exists an edge pointing from the target node of e_k to the source node of e_j. Similarly, the element $[(E^{\to})^p]_{kj}$ counts the number of ways that information is transferred from edge e_k to edge e_j through p nodes. The matrix exponential, $\exp(E^{\to})$, is a weighted sum of positive powers of $E^{\to}$, with transfers of information via many nodes having a smaller weight than transfers via few nodes; cf. (3.5). The matrix $E^{\to}$ generally is nonsymmetric.

Row k of $\exp(E^{\to})$ expresses the edge e_k in its broadcaster role, while column k of $\exp(E^{\to})$ expresses the role of edge e_k as a receiver. Specifically, a relatively large *edge line graph outcentrality*,

$${}^{e}\mathrm{LCout}_k = [\exp(E^{\to})\mathbf{1}]_k, \tag{3.7}$$

indicates that edge e_k is an important transmitter of information through the network, and a relatively large *edge line graph incentrality*,

$${}^{e}\mathrm{LCin}_k = [\exp(E^{\to T})\mathbf{1}]_k, \tag{3.8}$$

suggests that edge e_k is an important receiver of information.

4 Transformation of Node-Weighted to Edge-Weighted Graphs

In this section we consider ways of incorporating node weights into an adjacency matrix. Differently from edge weights, there is not a single natural approach that always can be used to encode node weights into an adjacency matrix. As will be seen below, different approaches may be useful in various circumstances. Subsection 4.1

describes available approaches, Subsection 4.2 discusses factorizable node weight functions, and Subsection 4.3 is concerned with the application of node weights via a line graph. The latter approach will be used in the node weighting method of this paper.

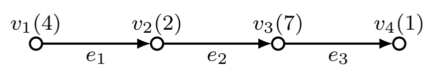
As shown in Subsection 2.2.2, the weighted adjacency matrix $\widetilde{A}$ of an edge-weighted graph can be defined in a natural by (2.2). We may rewrite this factorization as $\widetilde{A} = \widetilde{B}^{\mathrm{e}}\widetilde{B}^{\mathrm{i}T}$ by using the weighted incidence and exsurgence matrices $\widetilde{B}^{\mathrm{i}} = B^{\mathrm{i}}Z_1$ and $\widetilde{B}^{\mathrm{e}} = B^{\mathrm{e}}Z_2$, where the matrices Z_1 and Z_2 are diagonal with positive diagonal entries, and $Z = Z_2Z_1$ in the definition (2.2) of $\widetilde{A}$. Clearly, given the diagonal matrix Z, the choice of the diagonal matrices Z_1 and Z_2 is not unique.

Let the node weights $w_1, w_2, \ldots, w_n$ be given, and define the matrix $W = \mathrm{diag}(w_1, w_2, \ldots, w_n)$. We would like to encode these weights into an edge-weighted adjacency matrix $\widetilde{A}$. Hence, our goal is to determine edge weights

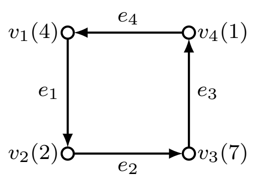
$$Z = \mathrm{diag}(z_1, z_2, \ldots, z_m) = H(W) \tag{4.1}$$

that depend only on W. In principle, each edge weight z_k may depend on all node weights, but in Subsections 4.1 and 4.2 we only consider "local" dependencies, i.e., each z_k is a function of the weights of the nodes at the endpoints of the edge only. We will consider "global" dependencies in Subsection 4.3.

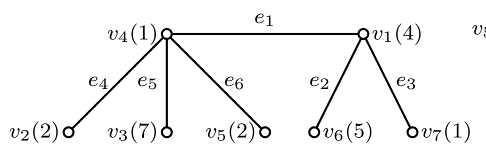
Below we discuss various ways of defining the function H in (4.1) and, for each approach, we describe a setting in which it is meaningful to use this approach. Often, methods already discussed in the literature turn out to be particular cases of weighting schemes of type (4.1). The graphs considered include the small node-weighted graphs in Figure 1. These graphs are well suited for comparison of the discussed modeling approaches. The node weights are displayed in parenthesis.



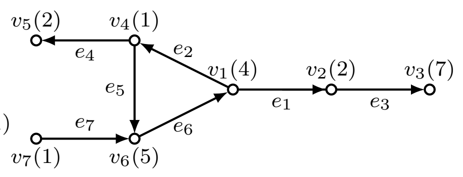
(a) A directed chain.



(b) A cyclic graph.



(c) An undirected graph.



(d) A directed graph.

Fig. 1 Node-weighted sample graphs. Node weights are displayed in parentheses.

4.1 Edge Weights from Endpoint Node Weights

Let the nodes v_i and v_j be the endpoints of the edge e_k, and let $z_k = h(w_i, w_j)$ for some function h. If the network is undirected, then h should be symmetric, i.e., $h(x, y) = h(y, x)$ for all $x, y \in \mathbb{R}$.

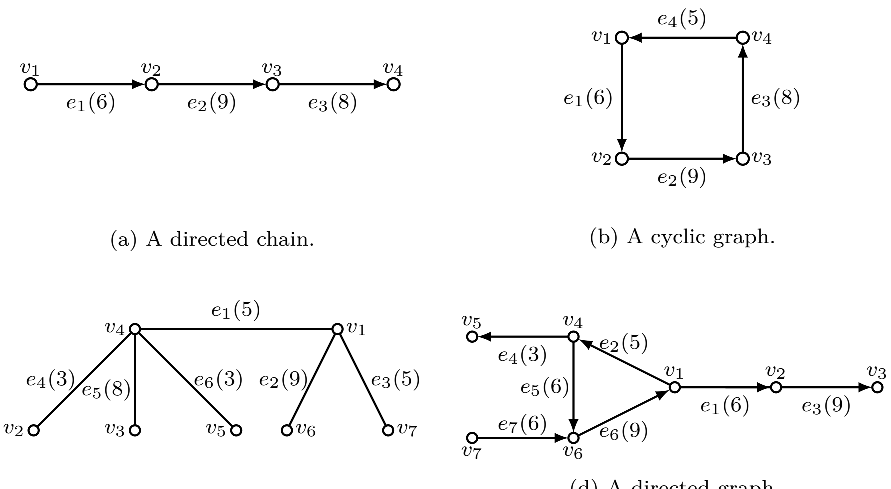
4.1.1 Sum of Endpoint Node Weights

Consider a network consisting of buildings as nodes, and each street linking two buildings as an edge. Each edge should be built large enough to accommodate all occupants from both buildings in case they have to escape a fire in the building they live in. If this network is node-weighted, with a node's weight proportional to the building capacity, then it is natural to convert the graph into an edge-weighted graph with each edge weight equal to the sum of its endpoint node weights. This weighting is most meaningful for undirected graphs. Thus, we determine Z from W by calculating the *sum node weights* (snw)

$$Z = \text{snw}(W) \quad \text{or} \quad z_k = h(w_i, w_j) = w_i + w_j. \tag{4.2}$$

This defines the function H in (4.1).

Figure 2 shows the edge-weighted graphs obtained by assigning each edge the sum of the weights from Figure 1 of the nodes it connects. Zou et al. [45] assigned to each edge half the sum of its endpoint node weights. This is simply a scaling by $\frac{1}{2}$ of the adjacency matrix that we obtain.



(a) A directed chain.

(b) A cyclic graph.

(c) An undirected graph.

(d) A directed graph.

Fig. 2 Edge weighted graphs obtained from the graphs in Figure 1 using (4.2).

4.1.2 Product of Endpoint Node Weights

Another approach to assign node weights is to make the edge weight proportional to each one of the two endpoint node weights. By symmetry considerations, the constant of proportionality should be the same for all edges. This yields the *product node weights* (pnw)

$$Z = \mathrm{pnw}(W) \quad \text{or} \quad z_k = h(w_i, w_j) = w_i w_j. \tag{4.3}$$

A situation when this approach to define the functions h and H in (4.1) is meaningful arises, for instance, when the nodes represent cities, the edges represent roads that connect the cities, and the traffic between the cities is assumed to be proportional to the populations of the cities. The node weight assignment (4.3) also is appropriate when the node weight corresponds to the probability of the node becoming "activated" at a given time. If different nodes get activated independently of each other, and an edge is activated when both its endpoint nodes are activated, then the product node weight provides the probability of activation of each edge.

4.1.3 Inheriting the Weight of an Endpoint Node

When the network is directed, it may be meaningful for the function h to be non-symmetric. Examples include

$$h(w_i, w_j) = w_i \quad \text{and} \quad h(w_i, w_j) = w_j. \tag{4.4}$$

These functions correspond to inheriting the weight of the source node, and inheriting the weight of the target node, respectively. These weight assignment approaches are considered in [36] and [23], respectively. These approaches make the edge weight proportional to the weight of the source or target nodes. In the edge activation scenario described above, the weightings (4.4) correspond to that an edge is activated whenever its source or target are activated.

4.2 Factorizable Node Weight Functions

When the function h can be factored

$$h(w_i, w_j) = h_1(w_i) h_2(w_j), \tag{4.5}$$

the relation (4.1) can be expressed with matrices in a simple manner. Assume that (4.5) holds, and let $H_1(W)$ be the diagonal matrix with the kth diagonal entry equal to $h_1(w_i)$, where v_i is the source node of edge e_k, for $k = 1, 2, \ldots, m$. Define the diagonal matrix $H_2(W)$ analogously for target nodes. Then $H(W) = H_1(W)H_2(W)$, and we obtain from (2.2) that

$$\widetilde{A} = B^{\mathrm{e}} Z B^{\mathrm{i}T} = B^{\mathrm{e}} H(W) B^{\mathrm{i}T} = B^{\mathrm{e}} H_1(W) H_2(W) B^{\mathrm{i}T}.$$

It is natural to introduce the weighted incidence matrices $\widetilde{B}^{\mathrm{e}} = B^{\mathrm{e}} H_1(W)$ and $\widetilde{B}^{\mathrm{i}} = B^{\mathrm{i}} H_2(W)$. Then $\widetilde{A} = \widetilde{B}^{\mathrm{e}} (\widetilde{B}^{\mathrm{i}})^T$.

We may let $h(x, y) = x^\alpha y^\beta$ for fixed $\alpha, \beta \in \mathbb{R}$. This weighting scheme includes some of the schemes described above as special cases. For instance, $\alpha = \beta = 1$

corresponds to (4.3), and $\alpha = 1, \beta = 0$ and $\alpha = 0, \beta = 1$ correspond to (4.4). Negative values of α or β may make sense in some modeling situations. For example, $\alpha = 1$ and $\beta = -1$ corresponds to the case when each edge weight is proportional to the weight of the source node and inversely proportional to the weight of the target node. An example of this could be a network, in which the nodes are countries, edges are military attacks, and the node weights measure military strength; edge weights can be obtained that correspond to the effectiveness of the attack, which would be proportional to the strength of the attacker and inversely proportional to the strength of the defender.

4.3 Graph Node Weights to Line Graph Edge Weights

Roughly, the roles of nodes and edges of a graph $\mathcal{G}$ are interchanged in the associated line graph $\mathcal{G}^*$; see Subsection 2.3. It is therefore natural to consider how node weights of $\mathcal{G}$ can be incorporated as edge weights in an adjacency matrix for $\mathcal{G}^*$.

4.3.1 Simple Weighting

By using the expressions that relate the adjacency matrix of $\mathcal{G}^*$ to the incidence matrix (or incidence matrices) for $\mathcal{G}$ (see Subsection 2.3), we obtain expressions for incorporating node weights W for $\mathcal{G}$ as edge weights into the adjacency matrix for $\mathcal{G}^*$.

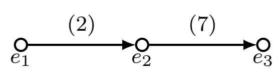
Consider first the directed case. Here we have that $E^{\rightarrow} = B^{\mathrm{i}T} B^{\mathrm{e}}$. Similarly to the expression for the weighted adjacency matrix $\widetilde{A}$ in (2.2), we define the *simply weighted* adjacency matrix of the line graph as

$$\widetilde{E}^{\rightarrow}_{\mathrm{SW}} = B^{\mathrm{i}T} W B^{\mathrm{e}}.$$

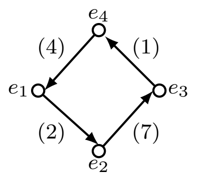
For undirected graphs, the unweighted adjacency matrix of the undirected line graph is $E = B^T B - 2I_m$, where B is the incidence matrix described in Subsection 2.2.1. We define the *simply weighted* adjacency matrix of the line graph as

$$\widetilde{E}_{\mathrm{SW}} = B^T W B - C,$$

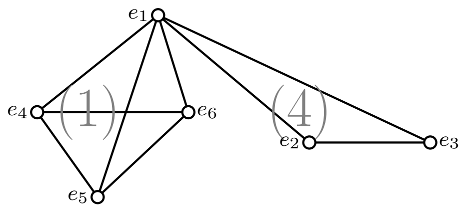
where C is the diagonal matrix with $c_{kk} = w_i + w_j$, whenever v_i and v_j are endpoints of the edge e_k, $k = 1, 2, \ldots, m$. Figure 3 shows edge-weighted line graphs corresponding to the graphs in Figure 1. In Figure 1(c), all connections in the left-hand side cluster have weight 1, and those in the right hand-side cluster have weight 4. We remark that this weighting method does not capture the weights of nodes that are only in direct contact with one edge in the original graph. In order to accommodate for these weights without changing the network topology, we add a self-loop to each node of the graphs in Figure 1, and then determine the associated line graph. We only illustrate this approach in Figure 4 for the graph in Figure 1(a), but we perform it on all graphs from this point onward. Note that, while these self-loops add nodes and edges in the line graph, we are not concerned with their ranking.



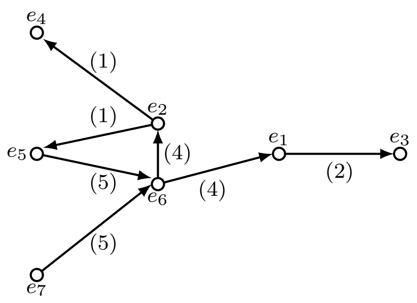
(a) A directed chain.



(b) A cyclic graph.

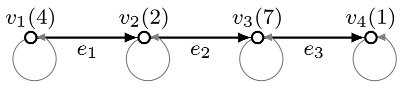


(c) An undirected graph.

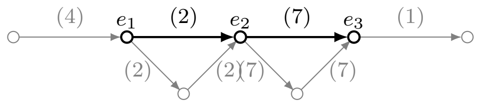


(d) A directed graph.

Fig. 3 Edge-weighted line graphs of the node-weighted graphs in Figure 1 sample networks.



(a) Adding self-loops to the graph of Figure 1(a).



(b) Edge-weighted line graph.

Fig. 4 Added self-loops to the graph in Figure 1(a) and the corresponding line graph. The effect of these additions is displayed in gray.

4.3.2 Scaling by Node Degree

A node v_i in $\mathcal{G}$ does not necessarily correspond to a single edge in $\mathcal{G}^*$. In fact, in undirected graphs, each node v_i produces a complete subgraph (a clique) in $\mathcal{G}^*$, containing $\binom{d_{v_i}}{2}$ edges in $\mathcal{G}^*$, where d_{v_i} denotes the degree of v_i in $\mathcal{G}$. The simple weighting approach described above assigns the weight w_i to all those $\binom{d_{v_i}}{2}$ edges in $\mathcal{G}^*$.

For example, in Figure 3(c), we notice that all edges of the cluster on the left-hand side have weight 1, which is the weight of node v_4 connecting these edges in Figure 1(c). From a modeling point of view, one may argue that in some applications, v_4 should distribute its weights to the surrounding edges, i.e., each edge should have the weight $\frac{1}{4}$. This suggests scaling the weights of those edges by the degree of v_4.

For an undirected graph, where D is the diagonal matrix holding the degree of its nodes, we define the *degree scaled* weighted adjacency matrix of the line graph as

$$\widetilde{E}_{\mathrm{DS}} = B^T W D^{-1} B - C_{\mathrm{DS}}, \tag{4.6}$$

where $C_{\rm DS} = [\text{diag}(c_{kk})]$ is the diagonal matrix with $c_{kk} = w_i/d_{v_i} + w_j/d_{v_j}$, and the nodes v_i and v_j, of degrees d_{v_i} and d_{v_j}, respectively, are the endpoints of the edge e_k in $\mathcal{G}$.

For directed networks, each node v_i in $\mathcal{G}$ results in indegree(v_i) × outdegree(v_i) edges in $\mathcal{G}^*$ (connecting each of the $\mathcal{G}$-edges inciding on v_i to each of the $\mathcal{G}$-edges exsurging from v_i). The number of edges in the line graph depends on the number of edges exsurging from the nodes in the original graph. Let $D_{\rm out}$ be the diagonal matrix holding the out-degrees of the nodes of $\mathcal{G}$ and define the *out-degree scaled* weighted adjacency matrix of the line graph as

$$\widetilde{E}^{\rightarrow}_{\rm ODS} = B^{\rm i}{}^T W D_{\rm out}^{-1} B^{\rm e}. \tag{4.7}$$

The *in-degree scaled* version, $\widetilde{E}^{\rightarrow}_{\rm IDS}$, is defined similarly.

4.3.3 Strong Degree Scaling

Rather than scaling node v_i by its degree d_{v_i}, it may in some situations be meaningful to divide by the number of the corresponding edges in $\mathcal{G}^*$. For undirected graphs, this means dividing by $\binom{d_{v_i}}{2}$. The algebra is similar to the scaling above, using a diagonal matrix $D_{\rm s}$ that contains the values $\binom{d_{v_i}}{2}$, $i = 1, 2, \ldots, n$, instead of the matrix D in (4.6). This gives the *strongly out-degree scaled* adjacency matrix $\widetilde{E}_{\rm SDS}$.

For directed networks, we similarly define the *strongly degree scaled* weighted adjacency matrix of the line graph as

$$\widetilde{E}^{\rightarrow}_{\rm SDS} = B^{\rm i}{}^T W D_{\rm s}^{-1} B^{\rm e}, \tag{4.8}$$

which is analogous to (4.7). To avoid division by zero when a node is a source (and therefore has zero indegree) or a sink (and then has zero outdegree), we add self-loops to each node of $\mathcal{G}$ before deriving $\mathcal{G}^*$. This gives the *strongly in-degree scaled* weighted adjacency matrix of the line graph, $\widetilde{E}^{\rightarrow}_{\rm SIDS}$.

We will illustrate the performance of severel of the weighted adjacency matrices of line graphs in computed examples in Section 6.

5 The Sensitivity of Node Centrality to Weight Change

This section is concerned with how the importance of a node changes when its weight is modified. In particular, we show that the rank of a node as broadcaster will increase, or at least remain the same, if its weight is increased. Related issues have been discussed by Bini et al. [8] for ranking methods based on the relative size of the entries of the left Perron vector of a row stochastic adjacency matrix of a graph.

To simplify notation, we denote the edge-weighted adjacency matrix by A in this section (this matrix is referred to as $\widetilde{A}$ elsewhere in this paper). Let the weight of node v_s increase. Thus, the edge(s) exsurging from node v_s increase in weight, while no edge exsurging from v_s decreases in weight. With each increment δ in the weight A_{st} of the edge pointing from node v_s to node v_t, the adjacency matrix associated with the perturbed graph can be expressed as

$$\widehat{A} = A + \delta \mathbf{1}_s \mathbf{1}_t^T,$$

where $\mathbf{1}_t = [0, \ldots, 0, 1, 0, \ldots, 0]^T \in \mathbb{R}^n$ denotes the t^{th} axis vector. We show in Section 5.2 that the ADR measure increases the most for node v_s (compared to all other nodes), under certain conditions on the graph. Therefore, its ranking as broadcaster either increases or remains the same.

5.1 Preliminaries

Lemma 1 *Let the nonnegative matrix $A \in \mathbb{R}^{n\times n}$ satisfy*

$$A\mathbf{1} \leq \mathbf{1},$$

where the inequality is component-wise. Thus, A is substochastic. Then for any integer $p \geq 1$,

$$A^p\mathbf{1} \leq \mathbf{1} \tag{5.1}$$

and

$$\exp(A)\mathbf{1} \leq e\mathbf{1}. \tag{5.2}$$

These inequalities are sharp.

Proof. The result is easily shown by induction. The inequality (5.1) holds for $p = 1$ by assumption. Assume that (5.1) holds for $p > 1$. Then

$$A^{p+1}\mathbf{1} = A(A^p\mathbf{1}) \leq A\mathbf{1} \leq \mathbf{1},$$

where we have used that all entries of A are nonnegative. For the exponential, we have

$$\exp(A)\mathbf{1} = \sum_{p=0}^{\infty} \frac{A^p\mathbf{1}}{p!} \leq \sum_{p=0}^{\infty} \frac{\mathbf{1}}{p!} = e\mathbf{1}.$$

The inequalities (5.1) and (5.2) become equalities for certain matrices, including the identity matrix and the cyclic shift matrix. The latter corresponds to an unweighted cyclic graph. □

Results analogous to those of Lemma 1 also hold for A^T.

Lemma 2 *Let the matrix A satisfy the conditions of Lemma 1. Then for any non-negative integer n_2,*

$$\sum_{n_1=1}^{\infty} \frac{(n_2+1)!}{(n_2+n_1+1)!} A^{n_1}\mathbf{1} < \mathbf{1}, \tag{5.3}$$

where the inequality holds component-wise.

Proof. We first bound the coefficients in (5.3) by induction over n_1. For $n_1 = 1$, we have

$$\frac{(n_2+1)!}{(n_2+2)!} = \frac{1}{n_2+2} \leq \frac{1}{2n_1!}.$$

Let $n_1 \geq 1$ be an arbitrary integer and assume that

$$\frac{(n_2+1)!}{(n_2+n_1+1)!} < \frac{1}{2n_1!}.$$

We would like to show that the above inequality holds for n_1 replaced by $n_1 + 1$. Using the above inequality, we obtain

$$\frac{(n_2+1)!}{(n_2+n_1+2)!} = \frac{(n_2+1)!}{(n_2+n_1+1)!(n_2+n_1+2)} < \frac{1}{2n_1!(n_2+n_1+2)}$$
$$< \frac{1}{2n_1!(n_1+1)} = \frac{1}{2(n_1+1)!}.$$

It follows that

$$\sum_{n_1=1}^{\infty} \frac{(n_2+1)!}{(n_2+n_1+1)!} A^{n_1}\mathbf{1} < \sum_{n_1=1}^{\infty} \frac{1}{2(n_1+1)!} A^{n_1}\mathbf{1} < \frac{1}{2}(\exp(A)\mathbf{1} - \mathbf{1}) \leq \frac{e-1}{2}\mathbf{1} < \mathbf{1},$$

where the penultimate inequality is a consequence of (5.2). This shows (5.3). □

5.2 Matrix Perturbation Results

This section considers adjacency matrices that satisfy the conditions of Lemma 1. As usual, $\mathbf{1} = [1,1,\ldots,1]^T \in \mathbb{R}^n$ is the vector of only ones, and $\mathbf{1}_j = [0,\ldots,0,1,0,\ldots,0]^T \in \mathbb{R}^n$ denotes the j^{th} axis vector for $j = 1,2,\ldots,n$. We will perturb the entry A_{st} of the adjacency matrix $A \in \mathbb{R}^{n\times n}$ by δ. This perturbation is denoted by $\delta A = \delta \mathbf{1}_s \mathbf{1}_t^T$. We will use the formulas

$$(\delta A)\mathbf{1} = \delta \mathbf{1}_s, \quad A(\delta A) = \delta(A\mathbf{1}_s\mathbf{1}_t^T), \quad (\delta A)A = \delta \mathbf{1}_s\mathbf{1}_t^T A.$$

When $s \neq t$, we have $(\delta A)^2 = 0$.

Theorem 1 *Let the adjacency matrix $A = [A_{ij}] \in \mathbb{R}^{n\times n}$ for the graph $\mathcal{G}$ satisfy the conditions of Lemma 1. Add $\delta > 0$ to the matrix entry A_{st} for some $s \neq t$, without changing any of the other entries of A. If δ is small enough, then the* ADR*-value of the vertex v_s increases more than the* ADR*-value of any other vertex. It follows that the rank of the vertex v_s as a broadcaster either increases or stays the same. More precisely, let $\delta A = \delta \mathbf{1}_s \mathbf{1}_t^T$ and $\widehat{A} = A + \delta A$. Then*

$$[\exp(\widehat{A})\mathbf{1}]_s - [\exp(A)\mathbf{1}]_s > [\exp(\widehat{A})\mathbf{1}]_q - [\exp(A)\mathbf{1}]_q \quad \forall q \neq s. \tag{5.4}$$

Proof. The binomial expansion gives

$$\exp(\widehat{A}) - \exp(A) = \sum_{p=0}^{\infty} \frac{1}{p!}\left((A+\delta A)^p - A^p\right) + \delta A \tag{5.5}$$

$$+ \sum_{p=2}^{\infty} \frac{1}{p!}\left(A^{p-1}(\delta A) + A^{p-2}(\delta A)A + \ldots + A(\delta A)A^{p-2} + (\delta A)A^{p-1}\right) + \mathcal{O}(\delta^2).$$

Multiplying the terms in the above sum by $\mathbf{1}$ from the right-hand side gives

$$\begin{aligned}
&\text{for } p=2, \quad \frac{1}{2!}(A(\delta A)+(\delta A)A)\mathbf{1} = \frac{\delta}{2!}(A\mathbf{1}_s + \mathbf{1}_s\mathbf{1}_t^T A\mathbf{1}),\\
&\text{for } p=3, \quad \frac{1}{3!}(A^2(\delta A)+A(\delta A)A+(\delta A)A^2)\mathbf{1} = \frac{\delta}{3!}(A^2\mathbf{1}_s + A\mathbf{1}_s\mathbf{1}_t^T A\mathbf{1} + \mathbf{1}_s\mathbf{1}_t^T A^2\mathbf{1}),\\
&\text{for } p=4, \quad \frac{1}{4!}(A^3(\delta A)+A^2(\delta A)A+A(\delta A)A^2+(\delta A)A^3)\mathbf{1} = \frac{\delta}{4!}(A^3\mathbf{1}_s + A^2\mathbf{1}_s\mathbf{1}_t^T A\mathbf{1}\\
&\qquad\qquad + A\mathbf{1}_s\mathbf{1}_t^T A^2\mathbf{1} + \mathbf{1}_s\mathbf{1}_t^T A^3\mathbf{1}),\\
&\qquad \dots .
\end{aligned}$$

Adding all the above terms "column-wise" and substituting into (5.5) multiplied by $\mathbf{1}$ from the right-hand side, we get

$$\begin{aligned}
\left(\exp(\widehat{A}) - \exp(A)\right)\mathbf{1} = \delta\Bigg(&\mathbf{1}_s + \frac{1}{2!}A\mathbf{1}_s + \frac{1}{3!}A^2\mathbf{1}_s + \frac{1}{4!}A^3\mathbf{1}_s + \dots\\
&+ \left(\frac{1}{2!}\mathbf{1}_s + \frac{1}{3!}A\mathbf{1}_s + \frac{1}{4!}A^2\mathbf{1}_s + \frac{1}{5!}A^3\mathbf{1}_s + \dots\right)(\mathbf{1}_t^T A\mathbf{1})\\
&+ \left(\frac{1}{3!}\mathbf{1}_s + \frac{1}{4!}A\mathbf{1}_s + \frac{1}{5!}A^2\mathbf{1}_s + \frac{1}{6!}A^3\mathbf{1}_s + \dots\right)(\mathbf{1}_t^T A^2\mathbf{1})\\
&+ \left(\frac{1}{4!}\mathbf{1}_s + \frac{1}{5!}A\mathbf{1}_s + \frac{1}{6!}A^2\mathbf{1}_s + \dots\right)(\mathbf{1}_t^T A^3\mathbf{1}) + \dots\Bigg) + \mathcal{O}(\delta^2)\\
= \delta \sum_{n_1=0}^{\infty}&\left(\sum_{n_2=0}^{\infty} \frac{A^{n_1}\mathbf{1}_s}{(n_1+n_2+1)!}(\mathbf{1}_t^T A^{n_2}\mathbf{1})\right) + \mathcal{O}(\delta^2).
\end{aligned}$$

It follows that

$$\begin{aligned}
[\exp(\widehat{A})\mathbf{1}]_s - [\exp(A)\mathbf{1}]_s &= \mathbf{1}_s^T\left(\exp(\widehat{A}) - \exp(A)\right)\mathbf{1}\\
&= \delta \sum_{n_1=0}^{\infty}\left(\sum_{n_2=0}^{\infty} \frac{\mathbf{1}_s^T A^{n_1}\mathbf{1}_s}{(n_1+n_2+1)!}(\mathbf{1}_t^T A^{n_2}\mathbf{1})\right) + \mathcal{O}(\delta^2) \qquad (5.6)\\
&\geq \delta \sum_{n_2=0}^{\infty} \frac{1}{(n_2+1)!}(\mathbf{1}_t^T A^{n_2}\mathbf{1}) + \mathcal{O}(\delta^2),
\end{aligned}$$

where the inequality is obtained by ignoring all terms with $n_1 > 0$. Now applying Lemma 2, and using the inequality $\mathbf{1}_q^T A^{n_1}\mathbf{1}_s \leq \mathbf{1}_q^T A^{n_1}\mathbf{1}$, and the fact that $\mathbf{1}_q^T A^0 \mathbf{1}_s = 0$ for $q \neq s$, gives

$$\begin{aligned}
\sum_{n_2=0}^{\infty} \frac{1}{(n_2+1)!}(\mathbf{1}_t^T A^{n_2}\mathbf{1}) &> \sum_{n_2=0}^{\infty}\left(\sum_{n_1=1}^{\infty} \mathbf{1}_q^T \frac{(n_2+1)!}{(n_2+n_1+1)!}A^{n_1}\mathbf{1}\right)\frac{1}{(n_2+1)!}(\mathbf{1}_t^T A^{n_2}\mathbf{1})\\
&\geq \sum_{n_2=0}^{\infty}\sum_{n_1=1}^{\infty} \frac{\mathbf{1}_q^T A^{n_1}\mathbf{1}_s}{(n_2+n_1+1)!}(\mathbf{1}_t^T A^{n_2}\mathbf{1}).
\end{aligned}$$

Comparing this expression and (5.6) shows that

$$[\exp(\widehat{A})\mathbf{1}]_s - [\exp(A)\mathbf{1}]_s > [\exp(\widehat{A})\mathbf{1}]_q - [\exp(A)\mathbf{1}]_q + \mathcal{O}(\delta^2).$$

This concludes the proof. □

Corollary 1 *Let the conditions of Theorem 1 hold. Assume in addition that the network consists of two clusters that are only connected by one directed edge from node v_s in the first cluster to node v_t in the second one. Then* (5.4) *holds for any* $\delta > 0$.

Proof. Let $\mathcal{G}_1$ be the graph of the first cluster of n_1 nodes, including the node v_s, and denote the associated adjacency matrix by $A_1 \in \mathbb{R}^{n_1 \times n_1}$. Similarly, let $\mathcal{G}_2$ be the graph of the second cluster of n_2 nodes, including the node v_t, and let $A_2 \in \mathbb{R}^{n_2 \times n_2}$ be the adjacency matrix of $\mathcal{G}_2$. We can arrange the rows and columns of A to have the form

$$A = \begin{bmatrix} A_1 & B_1 \\ 0 & A_2 \end{bmatrix}, \tag{5.7}$$

where B_1 is an $n_1 \times n_2$ matrix with all entries zero except for the entry (s,t), which holds the weight of the edge going from node v_s to node v_t. The lower left block of A is an $n_2 \times n_1$ matrix with all entries zero, because no edge goes from $\mathcal{G}_2$ to $\mathcal{G}_1$. We can show by induction that all powers of A have the structure

$$A^p = \begin{bmatrix} A_1^p & B_p \\ 0 & A_2^p \end{bmatrix},$$

where B_p is some $n_1 \times n_2$ matrix. Indeed, by (5.7), the matrix A^p has the desired structure for $p = 1$. Suppose that A^p has the desired structure. Then

$$\begin{aligned} A^{p+1} &= A^p A \\ &= \begin{bmatrix} A_1^p & B_p \\ 0 & A_2^p \end{bmatrix} \begin{bmatrix} A_1 & B_1 \\ 0 & A_2 \end{bmatrix} \\ &= \begin{bmatrix} A_1^{p+1} & A_1^p B_1 + B_p A_2 \\ 0 & A_2^{p+1} \end{bmatrix} \\ &= \begin{bmatrix} A_1^{p+1} & B_{p+1} \\ 0 & A_2^{p+1} \end{bmatrix}. \end{aligned}$$

It follows that $[A^p]_{ts} = 0$ for all $p = 1, 2, 3, \ldots$.

Let $\delta A = \delta \mathbf{1}_s \mathbf{1}_t^T$ with $s \neq t$. Then $(\delta A)^p = 0$ for all $p > 1$. In addition,

$$(\delta A) A^p (\delta A) = \delta \mathbf{1}_s \mathbf{1}_t^T A^p \delta \mathbf{1}_s \mathbf{1}_t^T = \delta^2 \mathbf{1}_s [A^p]_{ts} \mathbf{1}_t^T = 0.$$

Therefore, all terms of order $\mathcal{O}(\delta^2)$ in (5.5) vanish. This eliminates the need to require that $0 < \delta \ll 1$ in the proof of Theorem 1. $\square$

Corollary 2 *Let the conditions of Theorem 1 hold. Assume in addition that the graph $\mathcal{G}$ is such that there is no walk from node v_t to node v_s. Then* (5.4) *holds for any* $\delta > 0$.

Proof. Since there is no walk from node v_t to node v_s, we have $[A^p]_{ts} = 0$ for all $p = 1, 2, 3, \ldots$. Hence,

$$(\delta A) A^p (\delta A) = \delta \mathbf{1}_s \mathbf{1}_t^T A^p \delta \mathbf{1}_s \mathbf{1}_t^T = \delta^2 \mathbf{1}_s [A^p]_{ts} \mathbf{1}_t^T = 0.$$

Similarly to the proof of Corollary 1, we conclude that all terms of order $\mathcal{O}(\delta^2)$ in (5.5) vanish and the desired result follows. $\square$

Corollary 3 *Let the transpose of the adjacency matrix $A = [A_{ij}] \in \mathbb{R}^{n\times n}$ for the graph $\mathcal{G}$ satisfy the conditions of Lemma 1. Add $\delta > 0$ to the matrix entry A_{st} for some $s \neq t$, without changing any of the other entries of A. If δ is small enough, then the* AUR*-value of the vertex v_t increases more than the* AUR*-value of any other vertex. It follows that the rank of the vertex v_t as a receiver either increases or stays the same. More precisely, let $\delta A = \delta \mathbf{1}_s \mathbf{1}_t^T$ and $\widehat{A} = A + \delta A$. Then*

$$[\exp(\widehat{A}^T)\mathbf{1}]_t - [\exp(A^T)\mathbf{1}]_t > [\exp(\widehat{A}^T)\mathbf{1}]_q - [\exp(A^T)\mathbf{1}]_q \quad \forall q \neq t.$$

Proof. The result follows by applying Theorem 1 to the matrix A^T. □

5.3 Example of Sensitivity to Weight Change

Consider the weighted network in Figure 5. The numbers in parenthesis are edge weights. To satisfy the condition of Theorem 1, we scale the adjacency matrix A for the graph by the maximum of the largest column sum and the largest row sum, i.e., we divide all the entries of the adjacency matrix by 11.

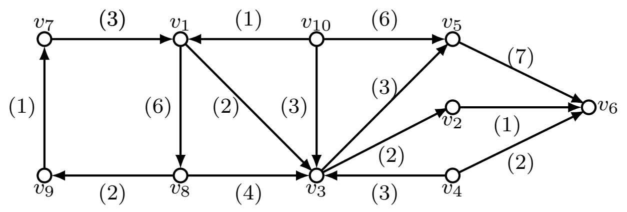


Fig. 5 Graph for the example in Section 5.3.

We increase the weight of the edge pointing from node v_1 to node v_3 by $\delta = 0.01$. The new adjacency matrix is $\widehat{A} = A + \delta \mathbf{1}_1 \mathbf{1}_3^T$. According to Theorem 1, node v_1 gets the highest ADR increase, which is in agreement with values reported in Table 1. By Corollary 3 no node should get a larger increase in its AUR-value than node v_3, which also is illustrated by Table 1.

Broadcasting nodes (ADR)		Receiving nodes (AUR)	
node v_q	$[\exp(\hat{A})\mathbf{1}]_q - [\exp(A)\mathbf{1}]_q$	node v_q	$[\exp(\hat{A}^T)\mathbf{1}]_q - [\exp(A^T)\mathbf{1}]_q$
v_1	2.395	v_3	2.168
v_7	0.299	v_2	0.280
v_{10}	0.100	v_5	0.280
v_8	0.008	v_6	0.074

Table 1 Ranking the top 4 nodes of the graph in Figure 5 showing the ADR- and AUR-values before and after perturbation of the edge pointing from node v_1 to node v_3. The graph $\mathcal{G}$ is scaled to satisfy the conditions of Lemma 1.

In the computed examples of the following sections, we scale the adjacency matrices A to be substochastic.

6 Computing Node Importance in Node-Weighted Graphs

Based on the several ways of incorporating node weights into adjacency matrices described in Section 4, we can use aggregate downstream and upstream reachability measures (described in Section 3) to find the most important nodes and edges in the graph. To rank the edges, we apply (3.6) (if the graph is undirected) and (3.7) and (3.8) (if the graph is directed). Which one of the line graphs from Section 4.3 ($\widetilde{E}^{\rightarrow}_{\mathrm{SW}}$, $\widetilde{E}^{\rightarrow}_{\mathrm{ODS}}$, $\widetilde{E}^{\rightarrow}_{\mathrm{SDS}}$, or their counterparts for undirected networks) is most appropriate depends on the application. Recall that we add self-loops before deriving the line graph. In analogy with (3.3) and (3.4), we refer to the edge ranking determined by (3.7) as the ability of edges to broadcast information, and the edge ranking obtained with (3.8) as the ability of edges to receive information. Thus, we rank edges as broadcasters and receivers, just like we rank nodes.

We apply (3.3) and (3.4) to rank the nodes and have to decide which edge-weighted adjacency matrix to use. The edge weights may be defined by the methods considered in Section 4. These methods were dictated by particular application. We now will describe two approaches that are independent of the application. They will be used in subsequent computed examples.

After ranking the edges of the graph via the line graph, we can substitute these values as edge weights (without the self-loops) into the original graph, i.e.,

$$Z = \{{}^{e}\mathrm{LC}\}_k \quad \text{from (3.6), if the graph is undirected,}$$

$$Z = \{{}^{e}\mathrm{LCout}\}_k + \{{}^{e}\mathrm{LCin}\}_k \quad \text{from (3.7) and (3.8), if the graph is directed.} \tag{6.1}$$

We define the edge-weighted adjacency matrix by (2.2). An advantage of these methods is that they determine the weight of each edge by taking into consideration not only the weights of both its source and target nodes, but also of all other nodes in the graph.

	Broadcasting edges (^{e}LCout)			Receiving edges (^{e}LCin)		
edge e_k	$[e^{\widetilde{E}^{\rightarrow}_{\mathrm{SW}}}\mathbf{1}]_k$	$[e^{\widetilde{E}^{\rightarrow}_{\mathrm{ODS}}}\mathbf{1}]_k$	$[e^{\widetilde{E}^{\rightarrow}_{\mathrm{SDS}}}\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\rightarrow}_{\mathrm{SW}}})^T\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\rightarrow}_{\mathrm{ODS}}})^T\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\rightarrow}_{\mathrm{SDS}}})^T\mathbf{1}]_k$
e_1	1.378	1.379	1.379	1.286	1.286	1.286
e_2	**2.146**	**2.167**	**1.317**	1.317	1.317	1.339
e_3	1.071	1.143	1.143	**2.214**	**2.214**	**2.218**

	Broadcasting nodes (ADR)			Receiving nodes (AUR)		
node v_i	SW	ODS	SDS	SW	ODS	SDS
v_1	2.275	2.270	2.270	1.000	1.000	1.000
v_2	**2.474**	**2.482**	**2.482**	1.769	1.765	1.765
v_3	1.949	1.964	1.959	2.385	2.382	2.382
v_4	1.000	1.000	1.000	**2.545**	**2.568**	**2.568**

Table 2 Top part: Ranking of edges of example (a) in Figure 1 as broacasters and receivers through the network using (3.7) and (3.8), for the adjacency matrix options of the line graph described in Section 4.3. The options are *simply weighted* (SW): $\widetilde{E}^{\rightarrow}_{\mathrm{SW}} = B^{\mathrm{i}T}WB^{\mathrm{e}}$, *out-degree scaled* (ODS): $\widetilde{E}^{\rightarrow}_{\mathrm{ODS}} = B^{\mathrm{i}T}WD_{\mathrm{out}}^{-1}B^{\mathrm{e}}$, and *strongly degree scaled* (SDS): $\widetilde{E}^{\rightarrow}_{\mathrm{SDS}} = B^{\mathrm{i}T}WD_{\mathrm{io}}^{-1}B^{\mathrm{e}}$. Bottom part: For the options above, the edges in Figure 1 are given the weight as the sum of broadcasting and receiving values, then ADR and AUR are calculated to rank nodes. Highest values are bold.

	Broadcasting edges (^{e}LCout)			Receiving edges (^{e}LCin)		
edge e_k	$[e^{\widetilde{E}^{\rightarrow}_{\text{SW}}}\mathbf{1}]_k$	$[e^{\widetilde{E}^{\rightarrow}_{\text{ODS}}}\mathbf{1}]_k$	$[e^{\widetilde{E}^{\rightarrow}_{\text{SDS}}}\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\rightarrow}_{\text{SW}}})^T\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\rightarrow}_{\text{ODS}}})^T\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\rightarrow}_{\text{SDS}}})^T\mathbf{1}]_k$
e_1	1.173	1.173	1.173	1.181	1.181	1.181
e_2	**1.073**	**1.070**	**1.070**	1.179	1.179	1.179
e_3	1.205	1.197	1.197	**1.066**	**1.066**	**1.066**
	Broadcasting nodes com. dist.			Receiving nodes com. dist.		
node v_i	SW	ODS	SDS	SW	ODS	SDS
v_1	2.275	2.270	2.270	1.000	1.000	1.000
v_2	**2.474**	**2.482**	**2.482**	1.769	1.765	1.765
v_3	1.949	1.964	1.959	2.385	2.382	2.382
v_4	1.000	1.000	1.000	**2.545**	**2.568**	**2.568**

Table 3 Top part: Ranking of edges of example (a) in Figure 1 as broacasters and receivers through the network using (3.7) and (3.8), for the adjacency matrix options of the line graph described in Section 4.3. The options are *simply weighted* (SW): $\widetilde{E}^{\rightarrow}_{\text{SW}} = B^{\text{i}T}WB^{\text{e}}$, *out-degree scaled* (ODS): $\widetilde{E}^{\rightarrow}_{\text{ODS}} = B^{\text{i}T}WD_{\text{out}}^{-1}B^{\text{e}}$, and *strongly degree scaled* (SDS): $\widetilde{E}^{\rightarrow}_{\text{SDS}} = B^{\text{i}T}WD_{\text{io}}^{-1}B^{\text{e}}$. Bottom part: For the options above, the edges in Figure 1 are given the weight as the sum of broadcasting and receiving values, then ADR and AUR are calculated to rank nodes. Highest values are bold.

We first illustrate this procedure for the graphs in Figure 1. Tables 2, 4, and 6 show the calculated measures for the components of the graphs in Figures 1(a), 1(b), and 1(d). The top part displays the values of $\{^e\text{LCout}\}_k$ and $\{^e\text{LCin}\}_k$ for each edge e_k, using the SW, ODS, and SDS adjacency matrix options of the line graph as described in Section 4.3.

To turn the original network into an edge-weighted one, we assign each edge a weight equal to the sum of its *in* and *out* values as in (6.1). The corresponding weighted adjacency matrix becomes $\widetilde{A} = B^{\text{e}}ZB^{\text{i}T}$. The bottom parts of Tables 2, 4, and 6 give the ADR- and AUR-values for each node v_i using the matrix $\widetilde{A}$. We have found that the rankings obtained are quite similar for all methods of Table 2. The computations are summarized by Algorithm 1.

Algorithm 1: Computing node importance for a node-weighted graph

```
1 Input: Graph $\mathcal{G}$ with $n$ nodes and a vector $w \in \mathbb{R}^n$ with node weights.
2 Output: Node ranking determined by the vector $\mathbf{w}_{\text{node\_ranking}}$.
3 Determine the incidence matrix $B^{\text{i}}$ and the exsurgence matrix $B^{\text{e}}$ for the graph $\mathcal{G}$,
    such that $A = B^{\text{e}}B^{\text{i}T}$ is the unweighted adjacency matrix for $\mathcal{G}$; cf. (2.1).
4 Determine the adjacency matrix $E$ for a line graph associated with the graph $\mathcal{G}$.
5 Incorporate the node weights for $\mathcal{G}$ as edge weights for the line graph. This gives a
    weighted adjacency matrix $\widetilde{E}$ for the line graph.
6 Evaluate $\mathbf{w}_{\text{edge\_weight}} = \exp(\widetilde{E})\mathbf{1}$ or $\mathbf{w}_{\text{edge\_weights}} = \exp(\widetilde{E}^T)\mathbf{1}$.
7 The vector $\mathbf{w}_{\text{edge\_weights}}$ contains edge weights for the graph $\mathcal{G}$.
8 Define the weighted adjacency matrix $\widetilde{A}$ given by (2.2) with $Z = \text{diag}(\mathbf{w}_{\text{edge\_weights}})$.
9 Compute $\mathbf{w}_{\text{node\_ranking}} = \exp(\widetilde{A})\mathbf{1}$ or $\mathbf{w}_{\text{node\_ranking}} = \exp(\widetilde{A}^T)\mathbf{1}$. The size of the
    entries of $\mathbf{w}_{\text{node\_ranking}}$ determines the ranking of the nodes of $\mathcal{G}$.
```

The edge-weighted adjacency matrix $\widetilde{A}$ in step 8 of Algorithm 1 is defined by (2.2) with $Z = \text{diag}(\mathbf{w}_{\text{node\_ranking}})$, i.e., the j^{th} diagonal entry of the diagonal matrix Z is the j^{th} component of the vector $\mathbf{w}_{\text{edge\_weights}}$. We remark that it is straightforward to replace the approaches of step 6 to determine edge weights $\mathbf{w}_{\text{edge\_weight}}$ by another measure, such as the subgraph centrality $\mathbf{w}_{\text{edge\_weight}} = \operatorname{diag}(\exp(\widetilde{E}))$. The approach used in step 6 has the advantage of being cheaper to evaluate; see Section 8 for a discussion on the evaluation of the expressions in step 6. Also the node ranking $\mathbf{w}_{\text{node\_ranking}}$ in step 9 can be determined as described in Section 8.

	Broadcasting edges (^{e}LCout)			Receiving edges (^{e}LCin)		
edge e_k	$[e^{\overrightarrow{\widetilde{E}}_{\mathrm{SW}}}\mathbf{1}]_k$	$[e^{\overrightarrow{\widetilde{E}}_{\mathrm{ODS}}}\mathbf{1}]_k$	$[e^{\overrightarrow{\widetilde{E}}_{\mathrm{SDS}}}\mathbf{1}]_k$	$[(e^{\overrightarrow{\widetilde{E}}_{\mathrm{SW}}})^T\mathbf{1}]_k$	$[(e^{\overrightarrow{\widetilde{E}}_{\mathrm{ODS}}})^T\mathbf{1}]_k$	$[(e^{\overrightarrow{\widetilde{E}}_{\mathrm{SDS}}})^T\mathbf{1}]_k$
e_1	1.3791	1.3791	1.3791	1.6389	1.6389	1.6389
e_2	**2.1713**	**2.1713**	**2.1713**	1.3419	1.3419	1.3419
e_3	1.1685	1.1685	1.1685	**2.2186**	**2.2186**	**2.2186**
e_4	1.6662	1.6662	1.6662	1.1871	1.1871	1.1871
	Broadcasting nodes (ADR)			Receiving nodes (AUR)		
node v_i	SW	ODS	SDS	SW	ODS	SDS
v_1	2.460	2.460	2.460	2.368	2.368	2.368
v_2	**2.647**	**2.647**	**2.647**	2.354	2.354	2.354
v_3	2.502	2.502	2.502	2.580	2.580	2.580
v_4	2.310	2.310	2.310	**2.619**	**2.619**	**2.619**

Table 4 Top part: Ranking of edges of example (b) in Figure 1 as broadcasters and receivers through the network using (3.7) and (3.8), for the *simply weighted* (SW), *out-degree scaled* (ODS), and *strongly degree scaled* (SDS) adjacency matrix options of the line graph as described in Section 4.3. Bottom part: For the options above, the edges in Figure 1 are given the weight as the sum of broadcasting and receiving values, then ADR and AUR are calculated to rank nodes. Highest values are bold.

Table 5 shows the calculated measures for the components of the graph in Figure 1(c).

	Edge centrality			Node centrality (ADR = AUR)			
edge e_k	$[e^{\widetilde{E}_{\mathrm{SW}}}\mathbf{1}]_k$	$[e^{\widetilde{E}_{\mathrm{DS}}}\mathbf{1}]_k$	$[e^{\widetilde{E}_{\mathrm{SDS}}}\mathbf{1}]_k$	node v_i	SW	DS	SDS
e_1	2.379	1.974	1.190	v_1	**2.335**	2.215	1.919
e_2	**2.437**	**2.360**	2.199	v_2	1.325	1.323	1.322
e_3	2.173	1.897	1.327	v_3	1.411	1.483	1.649
e_4	1.439	1.389	1.339	v_4	2.305	**2.280**	**2.250**
e_5	1.821	2.078	**2.705**	v_5	1.325	1.323	1.322
e_6	1.439	1.389	1.339	v_6	1.552	1.536	1.476
				v_7	1.492	1.431	1.287

Table 5 Left part: Ranking of edges of example (c) in Figure 1 using (3.7) and (3.8), for the adjacency matrix options of the line graph described in Section 4.3. The options are *simply weighted* (SW): $\widetilde{E}_{\mathrm{SW}} = B^TWB - C$, *degree scaled* (ODS): $\widetilde{E}_{\mathrm{DS}} = B^TWD^{-1}B - C_{DS}$, and *strongly degree scaled* (SDS): $\widetilde{E}_{\mathrm{SDS}} = B^TWD_s^{-1}B - C_{SDS}$. Right-hand side: For the given options, the edges in Figure 1 are given the weight as the centrality value from the left-hand side of this table, then ADR is calculated to rank nodes. Highest values are bold. Note that SW-values for node centrality are calculated after subtracting μI from $\widetilde{A}$, where μ is the spectral radius of $\widetilde{A}$, to avoid overflow.

	Dissipating edges (^{e}LCout)			Absorbing edges (^{e}LCin)		
edge e_k	$[e^{\widetilde{E}^{\to}_{\mathrm{SW}}}\mathbf{1}]_k$	$[e^{\widetilde{E}^{\to}_{\mathrm{ODS}}}\mathbf{1}]_k$	$[e^{\widetilde{E}^{\to}_{\mathrm{SDS}}}\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\to}_{\mathrm{SW}}})^T\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\to}_{\mathrm{ODS}}})^T\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\to}_{\mathrm{SDS}}})^T\mathbf{1}]_k$
e_1	1.308	1.341	1.341	1.728	1.474	1.232
e_2	1.237	1.159	1.078	1.728	1.474	1.232
e_3	1.467	**1.933**	**1.933**	1.321	1.306	1.290
e_4	1.133	1.267	1.267	1.158	1.100	1.047
e_5	1.888	1.830	1.775	1.158	1.100	1.047
e_6	**1.946**	1.609	1.292	**2.151**	**2.141**	**2.132**
e_7	1.888	1.830	1.775	1.067	1.067	1.067
	Dissipating nodes (ADR)			Absorbing nodes (AUR)		
node v_i	SW	ODS	SDS	SW	ODS	SDS
v_1	**2.374**	**2.306**	**2.202**	2.058	1.995	1.931
v_2	1.465	1.556	1.569	1.740	1.694	1.640
v_3	1.000	1.000	1.000	1.616	1.727	1.731
v_4	2.131	2.131	2.109	1.723	1.649	1.575
v_5	1.000	1.000	1.000	1.503	1.523	1.513
v_6	2.107	2.022	1.932	**2.161**	**2.144**	**2.128**
v_7	1.726	1.717	1.705	1.000	1.000	1.000

Table 6 Top part: Ranking of edges of example (d) in Figure 1 according to their importance as braodcasters and receivers using (3.7) and (3.8), for the *simply weighted* (SW), *out-degree scaled* (ODS), and *strongly degree scaled* (SDS) adjacency matrix options of the line graph as described in Section 4.3. Bottom part: For the options above, the edges in Figure 1 are given the weight as the sum of broadcasting and receiving values, then ADR and AUR are calculated to rank nodes. Highest values are bold.

7 Real-Life Examples

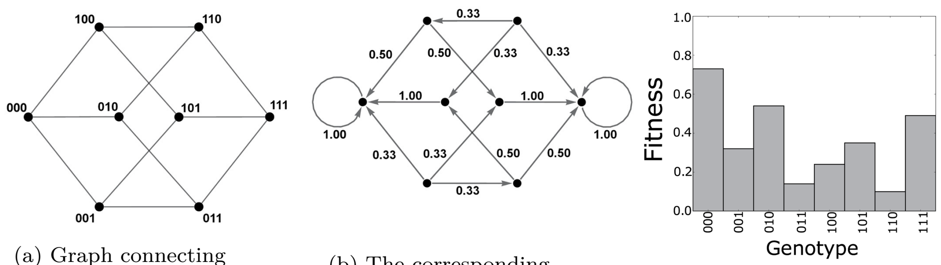
7.1 Genotype Mutation

This section discusses a biological example to illustrate some of the methods described above. To study the resistance of bacteria to an antibiotic, Nichol et al. [33] use an example that involves genotypes of 3 bits; see Figure 6(a). Each genotype is assigned a fitness level according to the fitness landscape in Figure 6(c). A genotype can only mutate to other genotypes if they have a higher fitness level. The authors present possible scenarios for the probability of these transitions, such as the probability being proportional to the fitness level increase, or the probability being that of a random walk as shown in Figure 6(b). In this paper we suggest a probability transition based on edge weights calculated using (6.1).

We display the network's node-weighted graph in Figure 7 with the node weights equal to the corresponding fitness levels. Note that we do not add self-loops to the nodes v_1 and v_8 as in [33], since we allow each genotype to remain the same, and not mutate, by adding self-loops to all nodes of the graph as described in Section 4.3.

We construct $\widetilde{E}^{\to}_{\mathrm{SW}}$, $\widetilde{E}^{\to}_{\mathrm{ODS}}$, and $\widetilde{E}^{\to}_{\mathrm{SDS}}$ from Section 4.3 to get the edge-weighted adjacency matrix of the line graph. We then calculate $^e\mathrm{LCout}_k$ and $^e\mathrm{LCin}_k$ from (3.7) and (3.8), respectively, and report them at the top of Table 7. The ODS method resulted in the edges e_3, e_4, and e_8 having the highest ^{e}LCout-value. Both the SW and SDS methods favor the edges e_5 and e_{10}. All methods ranked edge e_4 representing the transition from "010" to "000" to have the highest ^{e}LCin-value.

We use the sum (6.1) as the weight of edge e_k, for $k = 1, 2, \ldots, 12$, in the edge-weighted version of the graph in Figure 7. The importance of nodes as broadcasters



(a) Graph connecting genotypes.

(b) The corresponding stochastic transitions based on mutation to a better fitted neighbor.

(c) Fitness landscape.

Fig. 6 The genotype network for a bit strings of length 3 and the corresponding stochastic transitions according to the fitness levels and equations presented in [33].

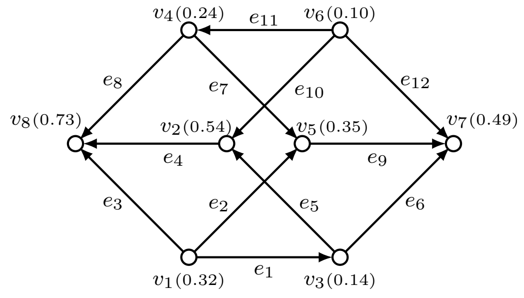


Fig. 7 The genotype directed graph of Figure 6. The nodes weights are from Figure 6(c).

is identified by calculating their ADR-values, and their importance as receivers by computing their AUR-values. These values are reported at the bottom of Table 7.

We conclude from the ADR-values of all methods considered that the genotypes “110”, i.e., node v_6, and “001”, i.e., node v_1, are the least stable genotype states, or the most likely to transition into another state. This is regardless of the choice of SW, ODS, or SDS. On the other hand, Table 7 also shows that genotype “111”, i.e., node v_7, is most likely to eventually be the last genotype reached by mutation, followed by “000”, i.e., node v_8. Note that the genotypes “111” and “000” have ADR-value 1 because, according to the fitness landscape in this example, they have higher fitness scores than the states that differ from them by one digit. Therefore “111” and “000” do not mutate. Similarly “110” and “001” have AUR-value 1, because they have lower fitness scores than the genotypes that differ from them by one digit, so the latter ones do not mutate to “110” or “001”. The specific value of 1 comes from the identity matrix in the Taylor expansion of the exponential function.

7.2 Social Networks: Medium and Twitter

In this example we investigate who are the most influential users in the Medium social network, based on their connectivity in Medium, and the influence of their

	Dissipating edges (^{e}LCout)			Absorbing edges (^{e}LCin)		
edge e_k	$[e^{\widetilde{E}^{\rightarrow}_{\text{SW}}}\mathbf{1}]_k$	$[e^{\widetilde{E}^{\rightarrow}_{\text{ODS}}}\mathbf{1}]_k$	$[e^{\widetilde{E}^{\rightarrow}_{\text{SDS}}}\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\rightarrow}_{\text{SW}}})^T\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\rightarrow}_{\text{ODS}}})^T\mathbf{1}]_k$	$[(e^{\widetilde{E}^{\rightarrow}_{\text{SDS}}})^T\mathbf{1}]_k$
e_1	1.221	1.070	1.035	1.146	1.037	1.006
e_2	1.351	1.172	1.172	1.146	1.037	1.006
e_3	1.333	**1.333**	**1.333**	1.146	1.037	1.006
e_4	1.333	**1.333**	**1.333**	**1.824**	**1.389**	**1.387**
e_5	**1.568**	1.276	1.276	1.135	1.043	1.021
e_6	1.224	1.224	1.224	1.135	1.043	1.021
e_7	1.351	1.172	1.172	1.228	1.074	1.037
e_8	1.333	**1.333**	**1.333**	1.228	1.074	1.037
e_9	1.224	1.224	1.224	1.536	1.251	1.248
e_{10}	**1.568**	1.276	1.276	1.046	1.011	1.002
e_{11}	1.379	1.120	1.060	1.046	1.011	1.002
e_{12}	1.224	1.224	1.224	1.046	1.011	1.002
	Dissipating nodes (ADR)			Absorbing nodes (AUR)		
node v_i	SW	ODS	SDS	SW	ODS	SDS
v_1	2.042	2.031	2.021	1.000	1.000	1.000
v_2	1.385	1.363	1.366	1.696	1.658	1.658
v_3	1.681	1.668	1.668	1.289	1.281	1.275
v_4	1.680	1.670	1.666	1.296	1.284	1.278
v_5	1.337	1.330	1.333	1.666	1.637	1.632
v_6	**2.051**	**2.036**	**2.030**	1.000	1.000	1.000
v_7	1.000	1.000	1.000	2.052	2.076	2.079
v_8	1.000	1.000	1.000	**2.177**	**2.162**	**2.162**

Table 7 Ranking edges and nodes of the genotype mutation graph in Figure 7 in dissipating and receiving through the network using (3.7) and (3.8), for the *simply weighted* (SW), *out-degree scaled* (ODS), and *strongly degree scaled* (SDS) adjacency matrix options of the line graph as described in Section 4.3. Bottom part: For the options above, the edges in Figure 1 are given the weight as the sum of dissipating and receiving values, then ADR and AUR are calculated to rank nodes. Highest values are bold.

Twitter accounts. We use a data set collected in 2016 that describes 1075983 users, who are identified by numerical IDs to protect their privacy [28]. The data set contains information about the users whom they follow and those they are followed by in Medium, along with information about whether their account is linked to their Twitter account, and some information about the Twitter account, if available. The data were collected to argue that linking Medium with Twitter is helpful to attract a large number of new users. We use the data provided publicly, and construct the network adjacency matrix from the edge list showing how accounts follow each other on Medium. We only take into account users who have a linked Twitter account, and use the number of followers they have on Twitter as node weights in the graph.

For computational purposes, we reduce our data set to users, who have more than 5000 Twitter followers. Our network consists of 10077 users represented by nodes and of 992539 directed connections expressed by edges. Because of the large network size, the MATLAB function `expm` cannot be be used to compute the matrix exponential. In the computations for this section, we use iterative Krylov subspace methods described in Section 8 to approximate the matrix exponential.

For each one of the options $\widetilde{E}^{\rightarrow}_{\text{SW}}$, $\widetilde{E}^{\rightarrow}_{\text{ODS}}$, and $\widetilde{E}^{\rightarrow}_{\text{SDS}}$ from Section 4.3, we calculate approximations of $^e\text{LCout} = \exp(\widetilde{E}^{\rightarrow})\mathbf{1}$ and $^e\text{LCin} = \mathbf{1}^T\exp(\widetilde{E}^{\rightarrow})$ using the techniques described in Section 8. We use the sum (6.1) as the weight of edges in the edge-weighted version of the graph representing Medium network. The top 15 ranked users for Medium are reported in Table 8. The three methods give almost identical rankings. We observe a redundancy in IDs between influencing accounts and those influenced. A likely explanation is that high-impacting social media users have high impact on each other, as well.

Dissipating nodes (ADR)			Absorbing nodes (AUR)		
SW	ODS	SDS	SW	ODS	SDS
14745	14486	14486	722	722	722
14486	14745	14745	45	45	45
6602	6602	6602	2265	553	553
553	553	553	553	2265	2265
2639	2639	2639	540	579	579
2631	2631	2631	6385	540	540
6385	6385	6385	579	6385	6385
722	722	722	2681	2681	2681
540	540	540	297	297	297
2265	2265	2265	2631	2631	2631
2681	2681	2681	2326	2326	2326
2187	2187	2187	2187	2690	2690
17956	17956	17956	2690	2187	2187
2210	2210	2210	4566	4566	4566
21398	21398	21398	2806	2806	2806
2326	2784	2784	7650	7650	7650
2784	2326	2326	8340	8340	8340
8839	3670	3670	2255	2255	2255
3670	8839	8839	8058	8058	8058
14965	14965	14965	2264	2264	2264

Table 8 Top 15 ranked accounts in Medium social network based on the associated accounts influence on Twitter using the *simply weighted* (SW), *out-degree scaled* (ODS), and *strongly degree scaled* (SDS) adjacency matrix options of the line graph as described in Section 4.3.

8 Computational Aspects

This section discusses some computational aspects of how to evaluate $\exp(E^{\rightarrow})\mathbf{1}$ and related matrix functions for large networks. For small to medium-sized (square) matrices M, we can first evaluate $\exp(M)$ with the MATLAB function `expm` (provided that overflow does not occur), and then multiply the matrix $\exp(M)$ by the vector $\mathbf{1}$. Methods for evaluating $\exp(M)$ for small to medium-sized matrices are described by Higham [24]. However, when the matrix M is large, the evaluation of $\exp(M)$ is difficult for several reasons:

1. Adjacency matrices M that represent networks typically are sparse, but the matrix $\exp(M)$ generally is not. The memory requirement for computing and subsequently storing $\exp(M)$ may be substantial.
2. The computational effort required for evaluating $\exp(M)$ for a large matrix M may be prohibitive.

3. Overflow is more likely to take place when M is a large adjacency matrix, than when M is small.

Our models require that we first evaluate $\exp(E^{\to})\mathbf{1}$ and $\mathbf{1}^T\exp(E^{\to})$ to form the edge-weighted graph. This defines the adjacency matrix $\widetilde{A}$. Subsequently, we compute $\exp(\widetilde{A})\mathbf{1}$ and $\mathbf{1}^T\exp(\widetilde{A})$ to rank nodes. To simplify the discussion, we will let M denote either one of the matrices $E^{\to}$ and $\widetilde{A}$.

To avoid overflow, we can evaluate (an approximation of) the spectral radius μ of M and compute $\exp(M-\mu I)$ instead of $\exp(M)$. The replacement of M by $M-\mu I$ does not affect the relative importance of edges and nodes in the graph. This rescaling has also been used in [20].

Large memory requirements make it impossible to evaluate the exponential of the matrices from the Medium-Twitter example in Section 7.2 on a standard laptop computer. This difficulty can be circumvented by approximating $\exp(M)$ by a low-rank matrix that is determined by application of a few steps of the Arnoldi or nonsymmetric Lanczos processes. We will compare these methods.

8.1 The Arnoldi Process

Let $M\in\mathbb{R}^{n\times n}$ and $\mathbf{1}=[1,\dots,1]^T\in\mathbb{R}^n$. Application of $\ell\ll n$ steps of the Arnoldi process to the matrix A with initial vector $\mathbf{1}$ gives the Arnoldi decomposition

$$MW_\ell = W_\ell H_\ell + \mathbf{g}_\ell \mathbf{1}_\ell^T, \tag{8.1}$$

where the columns of the matrix $W_\ell=[\mathbf{w}_1,\mathbf{w}_2,\dots,\mathbf{w}_\ell]\in\mathbb{R}^{n\times\ell}$ form an orthonormal basis for the Krylov subspace $\mathbb{K}_\ell(M,\mathbf{1})=\text{span}\{\mathbf{w}_1,M\mathbf{w}_1,\dots,M^{\ell-1}\mathbf{w}_1\}$ and $\mathbf{w}_1=\mathbf{1}/\|\mathbf{1}\|$. Here $\|\cdot\|$ denotes the Euclidean vector norm. The matrix $H_\ell\in\mathbb{R}^{\ell\times\ell}$ is of upper Hessenberg form and $\mathbf{g}_\ell\in\mathbb{R}^n$ satisfies $W_\ell^T\mathbf{g}_\ell=\mathbf{0}$; details on the Arnoldi process can be found, e.g., in Meurant [30] and Saad [39, Chapter 6]. We assume that ℓ is small enough so that the decomposition (8.1) with the stated properties exists. This is the generic situation. The computation of this decomposition requires the evaluation of ℓ matrix-vector products with the matrix M. We approximate $\exp(M)\mathbf{1}$ by the right-hand side of

$$\exp(M)\mathbf{1}\approx W_\ell\exp(H_\ell)\mathbf{1}_1\|\mathbf{1}\|; \tag{8.2}$$

see, e.g., [6,26] for properties of this approximation method. For many adjacency matrices M, it suffices to let ℓ in the decomposition (8.1) be fairly small to obtain a good enough approximation of $\exp(M)\mathbf{1}$. This is illustrated below. When the matrix M is large and ℓ is fairly small, the dominating computational effort required to compute the left-hand side of (8.1) is the evaluation of ℓ matrix-vector products with M. In applications of interest to us, the matrix M generally is nonsymmetric. Then the computations have to be repeated with M replaced by M^T when an approximation of $\exp(M^T)\mathbf{1}$ also is desired

8.2 The Nonsymmetric Lanczos Process

Application of ℓ steps of the nonsymmetric Lanczos process to the matrix $M \in \mathbb{R}^{n\times n}$ with initial vector $\mathbf{1} = [1, 1, \ldots, 1]^T \in \mathbb{R}^n$ gives the Lanczos decompositions

$$\begin{aligned} MV_\ell &= V_\ell T_\ell + \delta_{\ell+1}\mathbf{v}_{\ell+1}\mathbf{1}_\ell^T, \\ M^T W_\ell &= W_\ell T_\ell^T + \beta_{\ell+1}\mathbf{w}_{\ell+1}\mathbf{1}_\ell^T, \end{aligned} \tag{8.3}$$

where the columns of the matrix $V_\ell = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\ell] \in \mathbb{R}^{n\times\ell}$ span the Krylov subspace $\mathbb{K}_\ell(M, \mathbf{v}_1) = \text{span}\{\mathbf{v}_1, M\mathbf{v}_1, \ldots, M^{\ell-1}\mathbf{v}_1\}$ with $\mathbf{v}_1 = \mathbf{1}/\|\mathbf{1}\|$, and the columns of the matrix $W_\ell = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_\ell] \in \mathbb{R}^{n\times\ell}$ span the Krylov subspace $\mathbb{K}_\ell(M^T, \mathbf{w}_1) = \text{span}\{\mathbf{w}_1, M\mathbf{w}_1, \ldots, M^{\ell-1}\mathbf{w}_1\}$ with $\mathbf{w}_1 = \mathbf{1}/\|\mathbf{1}\|$. The columns of the matrices V_ℓ and W_ℓ are biorthogonal, i.e., $V_\ell^T W_\ell = I_\ell$. Moreover, $V_\ell^T \mathbf{w}_{\ell+1} = \mathbf{0}$ and $W_\ell^T \mathbf{v}_{\ell+1} = \mathbf{0}$. It follows from (8.3) that

$$W_\ell M V_\ell = T_\ell.$$

The matrix $T_\ell \in \mathbb{R}^{\ell\times\ell}$ is tridiagonal. For details on the nonsymmetric Lanczos method, see, e.g. Saad [39, Chapter 7]. We assume that ℓ is small enough so that the decompositions (8.3) with the stated properties exists. How to proceed when this is not the case is discussed in [3]. The computation of the decomposition (8.3) requires ℓ matrix-vector product evaluations with M and with M^T. Analogously to (8.2), we use the approximation

$$\exp(M)\mathbf{1} \approx V_\ell \exp(T_\ell)\mathbf{1}_1\|\mathbf{1}\|.$$

In our experience with large-scale real-world networks, we found that a small number of steps, ℓ, with the nonsymmetric Lanczos algorithms typically was sufficient to render a quite accurate approximation of $\exp(M)\mathbf{1}$.

8.3 Approximations for the Medium-Twitter Example

This section discusses in detail applications of the Arnoldi and nonsymmetric Lanczos processes to the ranking of the nodes of the Medium-Twitter example of Section 7.2. We first apply ℓ steps of the Arnoldi or nonsymmetric Lanczos processes to one of the three matrices $\widetilde{E}_{\overrightarrow{\text{SW}}}$, $\widetilde{E}_{\overrightarrow{\text{ODS}}}$, and $\widetilde{E}_{\overrightarrow{\text{SDS}}}$, with initial vector $\mathbf{1} = [1, \ldots, 1]^T$. The purpose of these computations is to determine edge weights and form the weighted adjacency matrix $\widetilde{A}$ following (6.1). Subsequently, we apply the Arnoldi and nonsymmetric Lanczos processes to approximate $\exp(\widetilde{A})\mathbf{1}$ and $\mathbf{1}^T\exp(\widetilde{A})$ to rank the nodes of the original graph.

Since we are interested in the node ranking, we show the smallest number of iterations with the Arnoldi and nonsymmetric Lanczos processes, when applied to one of the matrices $\widetilde{E}_{\overrightarrow{\text{SW}}}$, $\widetilde{E}_{\overrightarrow{\text{ODS}}}$, and $\widetilde{E}_{\overrightarrow{\text{SDS}}}$ and to $\widetilde{A}$, required so that the ranking of the top 20 nodes does not change when carrying out more steps. While this "stopping criterion" is not practical to use for the Arnoldi and nonsymmetric Lanczos processes, it illustrates that only a fairly small number of steps are required to gain insight into the node ordering. We found this to be true for other real-world large-scale networks as well. Hence, the computations required for many real-world large-scale network problems are not very expensive. Table 9 reports results for the matrices $\widetilde{E}_{\overrightarrow{\text{SW}}}$, $\widetilde{E}_{\overrightarrow{\text{ODS}}}$, and $\widetilde{E}_{\overrightarrow{\text{SDS}}}$ in the top row of each "window". Of course, identical ranking does not imply identical ADR- and AUR-values. The table therefore also displays the error in

	Arnoldi				Nonsymmetric Lanczos			
	number of iterations	time (sec.)	ADR rel. error	AUR rel. error	number of iterations	time (sec.)	ADR rel. error	AUR rel. error
SW	16	34.5	2.2E-2	9.9E-3	8	33.5	5.8E+1	1.4E+3
	17	36.2	4.1E-3	2.0E-3	11	41.6	4.5E-3	8.6E-2
	20	40.6	2.1E-5	3.9E-5	14	52.1	9.1E-6	1.1E-4
ODS	10	20.9	1.8E-1	2.0E-1	6	27.9	1.5E-2	1.5E-2
	11	24.0	3.2E-2	1.3E-3	7	30.4	2.3E-3	2.3E-3
	20	39.6	8.7E-9	3.4E-7	14	32.7	6.1E-7	6.2E-7
SDS	8	3.7	1.1E-1	3.1E-2	5	3.0	8.1E-1	8.1E-1
	9	4.1	1.7E-3	9.0E-3	6	3.5	6.6E-3	6.6E-3
	20	10.8	2.1E-10	3.1E-9	14	7.5	2.7E-7	1.6E-7

Table 9 Comparison of the performance of Arnoldi and nonsymmetric Lanczos approximations when applied to the Medium social network using *simply weighted* (SW), *out-degree scaled* (ODS), and *strongly degree scaled* (SDS) adjacency matrices for the line graph.

these values as well as the errors achieved when the number of iterations is increased. The "exact values" are determined by carrying out 100 iterations with the Arnoldi and nonsymmetric Lanczos processes.

The computations were carried out on a Lenovo Ideapad 510 laptop computer with a 2.5 GHz Intel Core i7 processor and 6 GB 2133 MHz DDR4 memory using MATLAB. Each time reported in Table 9 is the average of 10 runs. The adjacency matrix $\widetilde{E}_{\overrightarrow{\mathrm{SW}}}$ has the largest number of nonvanishing entries, and the adjacency matrix $\widetilde{E}_{\overrightarrow{\mathrm{SDS}}}$ the least. We observe that the former matrix requires the largest number of iterations and the longest computing time to satisfy our "stopping criterion", and the latter matrix requires the smallest number of iterations and the shortest computing time. In this example, the Arnoldi process requires more iterations to satisfy the "stopping criterion" than the nonsymmetric Lanczos process, but the fact that each iteration with the latter requires two matrix-vector product evaluations, while the former only demands the evaluation of one matrix-vector product evaluation per iteration, results in that application of the Lanczos process does not always require less computing time than the Arnoldi process.

The matrix line graph adjacency matrices are about $10^6 \times 10^6$, whereas the matrix $\widetilde{A}$ is only about $10^4 \times 10^4$. The line graph adjacency matrices are very sparse. On average, the Arnoldi and nonsymmetric Lanczos processes applied to the approximation of $\exp(\widetilde{A})\mathbf{1}$ required 5 iterations each, computed in less than 0.1 seconds.

9 Conclusion

While the incorporation of edge weights into methods based on the adjacency matrix is fairly direct, incorporating node weights is less straightforward. Similarly, the computation of importance or centrality measures for nodes has been well studied, but the extension of matrix function methods to compute importance measures for edges also runs into some difficulties. In this article we described how matrix function methods can be combined with node weights, as well as for computing edge importance measures, by using line graphs.

We discussed several novel modeling approaches to determine the most important nodes in a graph with positive node weights. These methods are built upon the notions of aggregate reachability (downstream and upstream, for directed graphs), which use the matrix exponential applied to a weighted adjacency matrix for the network. We investigated several ways in which node weights can be incorporated into an adjacency matrix. While this paper focuses on the application of the matrix exponential, it is straightforward to replace this function by a resolvent. Discussions on the use of the resolvent can be found in, e.g., [18, 25, 32].

In Section 5 we studied analytically the sensitivity of some of the measures of importance to changes in the weight of a single edge. Those results show that the measures of importance considered behave as expected (that is, increasing the weight of a node does not decrease its importance), at least for small perturbations; we also found some conditions under which the expected behavior holds regardless of the magnitude of the perturbation.

As many networks of interest are large, the adjacency matrix of the network graph, and especially the adjacency matrix of the corresponding line graph, can be very large. In Subsections 8.1 and 8.2 we discussed how to use the Arnoldi and nonsymmetric Lanczos processes to make approximate computations feasible.

Acknowledgments

The authors would like to thank a referee for comments that lead to clarifications of the presentation. This work was supported in part by NSF grants DMS-1729509 and DMS-1720259.

References

1. S. Achard, R. Salvador, B. Whitcher, J. Suckling, and E.D. Bullmore. A resilient, low-frequency, small-world human brain functional network with highly connected association cortical hubs. *Journal of Neuroscience*, 26:63–72, 2006.
2. J. M. Aldous and R. J. Wilson. *Graphs and Applications: An Introductory Approach.* Springer, London, 2000.
3. Z. Bai, D. Day, and Q. Ye. ABLE: An adaptive block Lanczos method for non-hermitian eigenvalue problems. *SIAM Journal Matrix Analysis and Applications*, 20:1060–1082, 2009.
4. A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences*, 101(11):3747–3752, 2004.
5. T. Baskaran, F. Blöchl, T. Brück, and F. J. Theis. The Heckscher–Ohlin model and the network structure of international trade. *International Review of Economics & Finance*, 20:135–145, 2011.
6. B. Beckermann and L. Reichel. Error estimation and evaluation of matrix functions via the Faber transform. *SIAM Journal on Numerical Analysis*, 47:3849–3883, 2009.
7. M. Benzi and C. Klymko. Total communicability as a centrality measure. *Journal of Complex Networks*, 1(2):124–149, 2013.
8. D. A. Bini, G. M. Del Corso, and F. Romani. Evaluating scientific products by means of citation-based models: a first analysis and validation. *Electronic Transactions on Numerical Analysis*, 33:1–16, 2008.
9. W.-K. Chen. *Graph Theory and its Engineering Applications*, volume 5. World Scientific, Singapore, 1997.
10. X. Chu, Z. Zhang, J. Guan, and S. Zhou. Epidemic spreading with nonlinear infectivity in weighted scale-free networks. *Physica A: Statistical Mechanics and its Applications*, 390(3):471–481, 2011.
11. J. J. Crofts, E. Estrada, D. J. Higham, and A. Taylor. Mapping directed networks. *Electronic Transactions on Numerical Analysis*, 37:337–350, 2010.

12. O. De la Cruz Cabrera, M. Matar, and L. Reichel. Analysis of directed networks via the matrix exponential. *Journal of Computational and Applied Mathematics*, 355:182–192, 2019.
13. O. De la Cruz Cabrera, M. Matar, and L. Reichel. Edge importance in a network via line graphs and the matrix exponential. *Numerical Algorithms*, 83:807–832, 2020.
14. R. Diestel. *Graph Theory.* Springer, Berlin, 2000.
15. E. Estrada. Edge adjacency relationships and a novel topological index related to molecular volume. *Journal of Chemical Information and Computer Sciences*, 35:31–33, 1995.
16. E. Estrada. *The Structure of Complex Networks: Theory and Applications.* Oxford University Press, Oxford, 2012.
17. E. Estrada and N. Hatano. Communicability in complex networks. *Physical Review E*, 77:036111, 2011.
18. E. Estrada and D. J. Higham. Network properties revealed through matrix functions. *SIAM Review*, 52:696–714, 2010.
19. E. Estrada and G. Silver. Accounting for the role of long walks on networks via a new matrix function. *Journal of Mathematical Analysis and Applications*, 449:1581–1600, 2017.
20. C. Fenu, D. Martin, L. Reichel, and G. Rodriguez. Network analysis via partial spectral factorization and Gauss quadrature. *SIAM Journal on Scientific Computing*, 35:A2046–A2068, 2013.
21. C. Godsil and G. F. Royle. *Algebraic Graph Theory.* Springer, New York, 2013.
22. I. Gutman and E. Estrada. Topological indices based on the line graph of the molecular graph. *Journal of Chemical Information and Computer Sciences*, 36:541–543, 1996.
23. J. Heitzig, J. F. Donges, Y. Zou, N. Marwan, and J. Kurths. Node-weighted measures for complex networks with spatially embedded, sampled, or differently sized nodes. *The European Physical Journal B*, 85(1):38, 2012.
24. N. J. Higham. *Functions of Matrices: Theory and Computation.* SIAM, Philadelphia, 2008.
25. L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.
26. L. A. Knizhnerman. Calculation of functions of unsymmetric matrices using Arnoldi's method. *USSR Comput. Math. Math. Phys.*, 31(1):1–9, 1991.
27. I. X. Y. Leung, S. Y. Chan, P. Hui, and P. Lio. Intra-city urban network and traffic flow analysis from gps mobility trace. *arXiv preprint arXiv:1105.5839*, 2011.
28. F. Li, Y. Chen, R. Xie, F. Ben Abdesslem, and A. Lindgren. Understanding service integration of online social networks: A data-driven study. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 848–853. IEEE, 2018.
29. X. Liu, J. Bollen, M. L. Nelson, and H. Van de Sompel. Co-authorship networks in the digital library research community. *Information Processing & Management*, 41:1462–1480, 2005.
30. G. Meurant. *Computer Solution of Large Linear Systems.* Elsevier, Amsterdam, 1999.
31. M. E. J. Newman. Analysis of weighted networks. *Physical Review E*, 70(5):056131, 2004.
32. M. E. J. Newman. *Networks: An Introduction.* Oxford University Press, Oxford, 2010.
33. D. Nichol, P. Jeavons, A. G Fletcher, R. A. Bonomo, P. K. Maini, J. L. Paul, R. A. Gatenby, A. R. A. Anderson, and J. G. Scott. Steering evolution with sequential therapy to prevent the emergence of bacterial antibiotic resistance. *PLoS Computational Biology*, 11(9):e1004493, 2015.
34. T. Opsahl, F. Agneessens, and J. Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, 2010.
35. K. Park, K. Lee, S. Park, and H. Lee. Telecommunication node clustering with node compatibility and network survivability requirements. *Management Science*, 46:363–374, 2000.
36. M. Pelillo, K. Siddiqi, and S. W. Zucker. Many-to-many matching of attributed trees using association graphs and game dynamics. In *International Workshop on Visual Form*, pages 583–593. Springer, 2001.
37. J. B. Pereira-Leal, A. J. Enright, and C. A. Ouzounis. Detection of functional modules from protein interaction networks. *PROTEINS: Structure, Function, and Bioinformatics*, 54:49–57, 2004.
38. C. Pizzuti. Overlapped community detection in complex networks. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 859–866. ACM, 2009.

39. Y. Saad. *Iterative Methods for Sparse Linear Systems.* 2nd ed., SIAM, Philadelphia, 2003.
40. S. Scarsoglio, F. Laio, and L. Ridolfi. Climate dynamics: a network-based approach for the analysis of global precipitation. *PLoS One*, 8(8):e71129, 2013.
41. A. J. Scott. A programming model of an integrated transportation network. In *Papers of the Regional Science Association*, volume 19, pages 215–222. Springer, 1967.
42. K. Thulasiraman and M. N. S. Swamy. *Graphs: Theory and Algorithms.* Wiley, New York, 1992.
43. A. H. Y. Tong, G. Lesage, G. D. Bader, H. Ding, H. Xu, X. Xin, J. Young, G. F. Berriz, R. L. Brost, and M. Chang. Global mapping of the yeast genetic interaction network. *Science*, 303:808–813, 2004.
44. D. Wei, X. Deng, X. Zhang, Y. Deng, and S. Mahadevan. Identifying influential nodes in weighted networks based on evidence theory. *Physica A: Statistical Mechanics and its Applications*, 392(10):2564–2575, 2013.
45. F. Zou, X. Li, S. Gao, and W. Wu. Node-weighted Steiner tree approximation in unit disk graphs. *Journal of Combinatorial Optimization*, 18(4):342–349, 2009.