# Exposing Multi-Relational Networks to Single-Relational Network Analysis Algorithms[1]

Marko A. Rodriguez[a], Joshua Shinavier[b]

[a] *T-5 Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, New Mexico 87545*
[b] *Tetherless World Constellation, Rensselaer Polytechnic Institute, Troy, New York, 12180*

## Abstract

Many, if not most network analysis algorithms have been designed specifically for single-relational networks; that is, networks in which all edges are of the same type. For example, edges may either represent "friendship," "kinship," or "collaboration," but not all of them together. In contrast, a multi-relational network is a network with a heterogeneous set of edge labels which can represent relationships of various types in a single data structure. While multi-relational networks are more expressive in terms of the variety of relationships they can capture, there is a need for a general framework for transferring the many single-relational network analysis algorithms to the multi-relational domain. It is not sufficient to execute a single-relational network analysis algorithm on a multi-relational network by simply ignoring edge labels. This article presents an algebra for mapping multi-relational networks to single-relational networks, thereby exposing them to single-relational network analysis algorithms.

*Key words:* multi-relational networks, path algebra, network analysis

## 1. Introduction

Much of graph and network theory is devoted to understanding and analyzing single-relational networks (also known as directed or undirected unlabeled graphs). A single-relational network is composed of a set of vertices (i.e. nodes) connected by a set of edges (i.e. links) which represent relationships of a single type. Such networks are generally defined as $G = (V, E)$, where $V$ is the set of vertices in the network, $E$ is the set of edges in the network, and $E \subseteq (V \times V)$. For example, if $i \in V$ and $j \in V$, then the ordered pair $(i, j) \in E$ represents an edge from vertex $i$ to vertex $j$.[2] Ignoring the differences in the vertices that edges connect, all edges in $E$ share a single nominal, or categorical, meaning. For example, the meaning of the edges in a single-relational social network may be kinship, friendship, or collaboration, but not all of them together in the same representation as there is no way distinguish what the edges denote. Moreover, single-relational networks may be weighted, where $w : E \to \mathbb{R}$ is a function that maps each edge in $E$ to a real value. Weighted forms are still considered single-relational as all the edges in $E$ have the same meaning; the only difference is the "degree of meaning" defined by $w$.

The network, as a data structure, can be used to model many real and artificial systems. However, because a network representation of a system can be too complicated to understand directly, many algorithms have been developed to map the network to a lower dimensional space. For a fine review of the many popular network analysis algorithms in use today, refer to [1, 2]. Examples of such algorithms, to name a few of the more popularly used algorithms, include the the family of geodesic [3, 4, 5, 6], spectral [7, 8, 9], and community detection algorithms [10, 11, 12].

Most network analysis algorithms have been developed for single-relational networks as opposed to multi-relational networks. A multi-relational network is composed of two or more sets of edges between a set of vertices. A multi-relational network can be defined as $M = (V, \mathbb{E})$, where $V$ is the set of vertices in the network, $\mathbb{E} = \{E_1, E_2, \ldots, E_m\}$ is a family of edge sets in the network, and any $E_k \subseteq (V \times V) : 1 \leq k \leq m$. Each edge set in $\mathbb{E}$ has a particular nominal, or categorical, interpretation. For example, within the same network

---

[2]This article is primarily concerned with directed networks, as opposed to undirected networks, in which an edge is an unordered set of two vertices (e.g. {i,j}). However, the formalisms presented work with undirected networks.

$M$, $E_1, E_2 \in \mathbb{E}$ may denote "kinship" and "coauthorship," respectively.

The multi-relational network is not new. These structures have been used in various disciplines ranging from cognitive science and artificial intelligence [13] (e.g. semantic networks for knowledge representation and reasoning) to social [1] and scholarly [14, 15] modeling. Furthermore, the multi-relational network is the foundational data structure of the emerging Web of Data [16, 17]. While a multi-relational network can be used to represent more complicated systems than a single-relational network, unfortunately, there are many fewer multi-relational algorithms than single-relational algorithms. Moreover, there are many more software packages and toolkits to analyze single-relational networks. Multi-relational network analysis algorithms that do exist include random walk [18], unique path discovery [19], community identification [20], vertex ranking [21], and path ranking algorithms [22].

The inclusion of multiple relationship types between vertices complicates the design of network algorithms. In the single-relational world, with all edges being "equal," the executing algorithm need not be concerned with the meaning of the edge, but only with the existence of an edge. Multi-relational network algorithms, on the other hand, must take this information into account in order to obtain meaningful results. For example, if a multi-relational network contains two edge sets, one denoting kinship ($E_1 \in \mathbb{E}$) and the other denoting coauthorship ($E_2 \in \mathbb{E}$), then for the purposes of a scholarly centrality algorithm, kinship edges should be ignored. In this simple case, the centrality algorithm can be executed on the single-relational network defined by $G = (V, E_2)$. However, there may exist more complicated semantics that can only be expressed through path combinations and other operations. Thus, isolating single-relational network components of $M$ is not sufficient. As a remedy to this situation, this article presents an algebra for defining abstract paths through a multi-relational network in order to derive a single-relational network representing vertex connectivity according to such path descriptions. From this single-relational representation, all of the known single-relational network algorithms can be applied to yield meaningful results. Thus, the presented algebra provides a means of exposing multi-relational networks to single-relational network analysis algorithms.

## 2. Path Algebra Overview

This section provides and overview of the various constructs of the path algebra and primarily serves as a consolidated reference. The following sec-

tions articulate the use of the constructs summarized here.

The purpose of the presented algebra is to transform a multi-relational network into a "semantically-rich" single-relational network. This is accomplished by manipulating a three-way tensor representation of a multi-relational network. The result of the three-way tensor manipulation yields an adjacency matrix (i.e. a two-way tensor).[3] The resultant adjacency matrix represents a "semantically-rich" single-relational network. The "semantically-rich" aspect of the resultant adjacency matrix (i.e. the single-relational network) is determined by the algebraic path description used to manipulate the original three-way tensor (i.e. the multi-relational network). More formally, the multi-relational path algebra is an algebraic structure that operates on $n \times n$ adjacency matrix "slices" of a $n \times n \times m$ three-way tensor representation of a multi-relational network in order to generate a $n \times n$ path matrix. The generated $n \times n$ path matrix represents a "semantically-rich" single-relational network that can be subjected to any of the known single-relational network analysis algorithms. The path algebra is a matrix formulation of the grammar-based random walker framework originally presented in [18]. However, the algebra generates "semantically-rich" single-relational networks, as opposed to only executing random walk algorithms in a "semantically-rich" manner. In other words, the algebra is cleanly separated from the analysis algorithms that are ultimately applied to it. This aspect of the algebra makes it generally useful in many network analysis situations.

The following list itemizes the various elements of the algebra to be discussed in §3.

- $\mathcal{A} \in \{0,1\}^{n \times n \times m}$: a three-way tensor representation of a multi-relational network.[4]

- $\mathbf{Z} \in \mathbb{R}_+^{n \times n}$: a path matrix derived by means of operations applied to $\mathcal{A}$.

- $\mathbf{R}_i \in \{0,1\}^{n \times n}$: a row "from" path filter.

- $\mathbf{C}_i \in \{0,1\}^{n \times n}$: a column "to" path filter.

---

[3]The term *tensor* has various meanings in mathematics, physics, and computer science. In general, a tensor is a structure which includes and extends the notion of scalar, vector, and matrix. A zero-way tensor is a scalar, a one-way tensor is a vector, a two-way tensor is a matrix, and a three-way tensor is considered a "cube" of scalars. For a three-way tensor, there are three indices used to denote a particular scalar value in the tensor.

[4]This article is primarily concerned with boolean tensors. However, note that the presented algebra works with tensors in $\mathbb{R}_+^{n \times n \times m}$.

- $\mathbf{E}_{i,j} \in \{0,1\}^{n \times n}$: an entry path filter.
- $\mathbf{I} \in \{0,1\}^{n \times n}$: the identity matrix as a self-loop filter.
- $\mathbf{1} \in 1^{n \times n}$: a matrix in which all entries are 1.
- $\mathbf{0} \in 0^{n \times n}$: a matrix in which all entries are 0.

The following list itemizes the various operations of the algebra to be discussed in §4.

- $\mathbf{A} \cdot \mathbf{B}$: ordinary matrix multiplication determines the number of $(\mathbf{A}, \mathbf{B})$-paths between vertices.
- $\mathbf{A}^\top$: matrix transpose inverts path directionality.
- $\mathbf{A} \circ \mathbf{B}$: Hadamard, entry-wise multiplication applies a filter to selectively exclude paths.
- $n(\mathbf{A})$: *not* generates the complement of a $\{0,1\}^{n \times n}$ matrix.
- $c(\mathbf{A})$: *clip* generates a $\{0,1\}^{n \times n}$ matrix from a $\mathbb{R}_+^{n \times n}$ matrix.
- $v^{\pm}(\mathbf{A})$: *vertex* generates a $\{0,1\}^{n \times n}$ matrix from a $\mathbb{R}_+^{n \times n}$ matrix, where only certain rows or columns contain non-zero values.
- $\lambda \mathbf{A}$: scalar multiplication weights the entries of a matrix.
- $\mathbf{A} + \mathbf{B}$: matrix addition merges paths.

In short, any abstract path $\sigma$ is a series of operations on $\mathcal{A}$ that can be generally defined as

$$\sigma : \{0,1\}^{n \times n \times m} \rightarrow \mathbb{R}_+^{n \times n}.$$

The resultant $\mathbb{R}_+^{n \times n}$ matrix is an adjacency matrix. Thus, the resultant matrix is a single-relational network. This resultant single-relational network can be subjected to single-relational network analysis algorithms while still preserving the semantics of the original multi-relational network.

Throughout the remainder of this article, scholarly examples are provided in order to illustrate the application of the various elements and operations of the path algebra. All of the examples refer to a single scholarly tensor denoted $\mathcal{A}$. The following list itemizes the tensor "slices" and their domains and ranges, where $H \subset V$ is the set of all humans, $A \subset V$ is the set of all articles, $J \subset V$ is the set of all journals, $S \subset V$ is the set of all subject categories, and $P \subset V$ is the set of all software programs:

- $\mathcal{A}^1$: `authored` : $H \rightarrow A$

- $\mathcal{A}^2$: `authoredBy` $: A \to H$
- $\mathcal{A}^3$: `cites` $: A \to A$
- $\mathcal{A}^4$: `contains` $: J \to A$
- $\mathcal{A}^5$: `category` $: J \to S$
- $\mathcal{A}^6$: `developed` $: H \to P$.

## 3. Path Algebra Elements

This section introduces the elements of the algebra and the next section articulates their use in the various operations of the algebra. The elements of the path algebra are structures that are repeatedly used when mapping a multi-relational tensor to a single-relational path matrix.

### 3.1. Three-way Tensor Representation of a Multi-Relational Network

A single-relational network defined as

$$G = (V, E \subseteq (V \times V))$$

can be represented as the adjacency matrix $\mathbf{A}$, where

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

without loss of information. This adjacency matrix is also known as a two-way tensor because it has two dimensions, each with an order of $n$, where $n = |V|$. Stated another way, $\mathbf{A} \in \{0,1\}^{n \times n}$.

A three-way tensor can be used to represent a multi-relational network [23]. If

$$M = (V, \mathbb{E} = \{E_1, E_2, \ldots, E_m \subseteq (V \times V)\})$$

is a multi-relational network, then

$$\mathcal{A}_{i,j}^k = \begin{cases} 1 & \text{if } (i,j) \in E_k : 1 \leq k \leq m \\ 0 & \text{otherwise.} \end{cases}$$

In this formulation, two dimensions have an order of $n$ while the third has an order of $m$, where $m = |\mathbb{E}|$. Thus, $\mathcal{A} \in \{0,1\}^{n \times n \times m}$ and any adjacency matrix "slice" $\mathcal{A}^k \in \{0,1\}^{n \times n} : 1 \leq k \leq m$. $\mathcal{A}$ represents the primary structure by which individual adjacency matrices are indexed and composed in order to derive a resultant single-relational path matrix.

### 3.2. Path Matrices

The path algebra operates on $n \times n$ adjacency matrix elements of $\mathcal{A}$ to construct a "semantically-rich" path matrix. The simplest path is made up of a single edge type. Thus, $\mathcal{A}^k$ is a path matrix for vertex-to-vertex paths of length 1. The meaning of that path is simply defined by the meaning of $\mathbb{E}_k$. When constructing complex paths through $\mathcal{A}$ (when utilizing multiple edge types in $\mathcal{A}$), the resulting matrix may have entries with values greater than 1. Furthermore, in conjunction with the use of $\mathbb{R}_+$ scalar multiplication (described in §4.3), entries of a path matrix may be non-negative real numbers. Therefore, in general, all path matrices discussed throughout the remainder of this article are matrices in $\mathbb{R}_+^{n \times n}$ and are denoted $\mathbf{Z}$. In short, the resultant $\mathbf{Z}$ matrix can be seen as a positively weighted single-relational network. In other words, $\mathbf{Z}$ denotes a single-relational network of the form $G = (V, E, w)$, where $w : E \rightarrow R_+$.

### 3.3. Filter Matrices

Filters are used to ensure that particular paths through $\mathcal{A}$ are either included or excluded from a path composition. Generally, a filter is a $\{0, 1\}^{n \times n}$ matrix. Filters may be derived from intermediate path matrices (described in §4.2) or may target a specific vertex that is known *a prior*. A vertex-specific filter is either a row $\mathbf{R}_i \in \{0, 1\}^{n \times n}$, column $\mathbf{C}_i \in \{0, 1\}^{n \times n}$, or entry $\mathbf{E}_{i,j} \in \{0, 1\}^{n \times n}$ filter.

1. A row filter is denoted $\mathbf{R}_i$, where all entries in row $i$ are equal to 1 and all other entries are equal to 0. Row filters are useful for allowing only those paths that have an origin of $i$.
2. A column filter is denoted $\mathbf{C}_i$, where all entries in column $i$ are equal to 1 and other entries are equal to 0. Column filters are useful for allowing only those paths that have a destination of $i$.
3. Entry filters are denoted $\mathbf{E}_{i,j}$ and have a 1 at the $(i, j)$-entry and 0 elsewhere. Entry filters allow only those paths that have an origin of $i$ or a destination of $j$.

Useful properties of the vertex-specific filters include:

- $\mathbf{R}_i = \mathbf{C}_i^\top$
- $\mathbf{C}_i = \mathbf{R}_i^\top$
- $\mathbf{E}_{i,j} = \mathbf{E}_{j,i}^\top$.

7

The identity matrix $\mathbf{I}$ is useful for allowing or excluding self-loops. Finally, the filter $\mathbf{1} \in 1^{n \times n}$ is a matrix in which all entries are equal to 1, and $\mathbf{0} \in 0^{n \times n}$ is a matrix in which all entries are equal to 0.

## 4. Path Algebra Operations

The previous section defined the common elements of the path algebra. These elements are composed with one another to create a path composition. This section discusses the various operations that are used when composing the aforementioned elements.

### 4.1. The Traverse Operation

A useful property of the single-relational adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ is that when it is raised to the $t^{\text{th}}$ power, the entry $\mathbf{A}_{i,j}^{(t)}$ is equal to the number of paths of length $t$ that connect vertex $i \in V$ to vertex $j \in V$ [24]. This is simple to prove using induction. Given, by definition, that $\mathbf{A}_{i,j}^{(1)}$ (i.e. $\mathbf{A}_{i,j}$) represents the number of paths that go from $i$ to $j$ of length 1 (i.e. a single edge) and by the rules of ordinary matrix multiplication,

$$\mathbf{A}_{i,j}^{(t)} = \sum_{l \in V} \mathbf{A}_{i,l}^{(t-1)} \cdot \mathbf{A}_{l,j} : t \geq 2.$$

The same mechanism for finding the number of paths of length $t$ in a single-relational network can be used to find the number of semantically meaningful paths through a multi-relational network. For example, suppose $\mathcal{A}^1$ has the label `authored`, $\mathcal{A}^2$ has the label `authoredBy`, $\mathcal{A}^3$ has the label `cites`, and

$$\mathbf{Z} = \mathcal{A}^1 \cdot \mathcal{A}^3 \cdot \mathcal{A}^2.$$

Semantically, $\mathbf{Z}_{i,j}$ is the number of paths from vertex $i$ to vertex $j$ such that a path goes from author $i$ to one the articles he or she has authored, from that article to one of the articles it cites, and finally, from that cited article to its author $j$.[5] The meaning of $\mathbf{Z}$ is `hasCited` $: H \to H$ and represents an edge if some author has cited some other author by means of their respective articles (i.e. an author citation network). This method is analogous to raising

---

[5]If vertex $i$ is not an author, then such a path composition would yield 0 paths from $i$ to any $j$. A path composition must respect the domains and ranges of the edge types if a meaningful path matrix is to be generated.

an adjacency matrix to the $t^{\text{th}}$ power, except that, in place of multiplying the same adjacency matrix with itself, a sequence of different adjacency matrix "slices" in $\mathcal{A}$ are used to represent typed paths with a compositionally defined meaning.

It is worth noting that ordinary matrix multiplication is not commutative. Thus, for the most part, when $\mathbf{A} \neq \mathbf{B}$, $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$. Given paths through a multi-relational network, this makes intuitive sense. The path from `authored` to `cites` is different than the path from `cites` to `authored`. In the first case, if the resultant path is seen as a mapping, then `authored` $: H \rightarrow A$ (i.e. human to article) and `cites` $: A \rightarrow A$ (i.e. article to article). Thus, through composition `cites` $\circ$ `authored` $: H \rightarrow A$.[6] However, in the latter case, composition is not possible as `cites` has a range of an article and `authored` has a domain of human.

Finally, any $n \times n$ adjacency matrix element of $\mathcal{A}$ can be transposed in order to traverse paths in the opposite direction. For example, given $\mathcal{A}_1$ and $\mathcal{A}_2$ identified as `authored` and `authoredBy`, respectively, where $\mathcal{A}^{1^\top} = \mathcal{A}^2$, then $\mathbf{Z} = \mathcal{A}^1 \cdot \mathcal{A}^3 \cdot \mathcal{A}^2 = \mathcal{A}^1 \cdot \mathcal{A}^3 \cdot \mathcal{A}^{1^\top}$. Thus, inverse edge types can be created using matrix transpose.

*4.2. The Filter Operation*

In many cases, it is important to exclude particular paths when traversing through $\mathcal{A}$. Various path filters can be defined and applied using the entry-wise Hadamard matrix product denoted $\circ$ [25], where

$$\mathbf{A} \circ \mathbf{B} = \begin{bmatrix} \mathbf{A}_{1,1} \cdot \mathbf{B}_{1,1} & \cdots & \mathbf{A}_{1,m} \cdot \mathbf{B}_{1,m} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{n,1} \cdot \mathbf{B}_{n,1} & \cdots & \mathbf{A}_{n,m} \cdot \mathbf{B}_{n,m} \end{bmatrix}.$$

The following list itemizes various properties of the Hadamard product:

- $\mathbf{A} \circ \mathbf{1} = \mathbf{A}$
- $\mathbf{A} \circ \mathbf{0} = \mathbf{0}$
- $\mathbf{A} \circ \mathbf{B} = \mathbf{B} \circ \mathbf{A}$
- $\mathbf{A} \circ (\mathbf{B} + \mathbf{C}) = (\mathbf{A} \circ \mathbf{B}) + (\mathbf{A} \circ \mathbf{C})$

---

[6]The symbol $\circ$ is overloaded in this article meaning both function composition and the Hadamard matrix product. The context of the symbol indicates its meaning.

- $\mathbf{A} \circ \lambda \mathbf{B} = \lambda(\mathbf{A} \circ \mathbf{B})$
- $\mathbf{A}^\top \circ \mathbf{B}^\top = (\mathbf{A} \circ \mathbf{B})^\top$.

If $\mathbf{A}$ is a $\{0,1\}^{n \times n}$ matrix, then $\mathbf{A} \circ \mathbf{A} = \mathbf{A}$. For the row, column, and entry filters,

- $\mathbf{R}_i \circ \mathbf{R}_j = \mathbf{0} \: : \: i \neq j$
- $\mathbf{C}_i \circ \mathbf{C}_j = \mathbf{0} \: : \: i \neq j$
- $\mathbf{R}_i \circ \mathbf{C}_j = \mathbf{E}_{i,j}$.

Finally, if $\mathbf{Z} \in \mathbb{R}_+^{n \times n}$ and $\mathbf{Z}$ has a trace of 0 (i.e. no self-loops), then $\mathbf{Z} \circ \mathbf{I} = \mathbf{0}$.

The Hadarmard product is used in the path algebra to apply a filter. As stated previously, a typical filter is a $\{0,1\}^{n \times n}$ matrix, where 0 entries set the corresponding path counts in $\mathbf{Z}$ to 0. The following subsections define and illustrate some useful functions to generate filters.

### 4.2.1. The Not Function

The *not* function is defined as

$$n : \{0,1\}^{n \times n} \rightarrow \{0,1\}^{n \times n}$$

with a function rule of

$$n(\mathbf{A}) = \mathbf{1} - \mathbf{A}.$$

In words, the *not* takes a $\{0,1\}^{n \times n}$ matrix and replaces all the 0s with 1s and all the 1s with 0s. Some evident and useful properties of a *not* filter include

- $n(n(\mathbf{A})) = \mathbf{A}$
- $\mathbf{A} \circ n(\mathbf{A}) = \mathbf{0}$
- $n(\mathbf{A}) \circ n(\mathbf{A}) = n(\mathbf{A})$.

Furthermore, if $\mathbf{Z} \in \mathbb{R}_+^{n \times n}$ and $\mathbf{Z}$ has a trace of 0, then $\mathbf{Z} \circ n(\mathbf{I}) = \mathbf{Z}$.

A *not* function is useful for excluding a set of paths to or from a vertex. For example, when constructing a coauthorship path matrix where $\mathcal{A}^1$ represents `authored`, the operation $\mathcal{A}^1 \cdot \mathcal{A}^{1\top} \circ n(\mathbf{I})$ will ensure that the `authored` relationship is taken and then the transpose of `authored` (i.e. `authoredBy`) is taken. However, if only these two operations are applied, then this does not yield a coauthorship matrix, as the traversal returns to the originating vertex (i.e. vertex $i$ is considered a coauthor of vertex $i$). Thus, the applied *not*-identity filter will remove all paths back to the source vertex, at which point a coauthorship path matrix is generated.

*4.2.2. The Clip Function*

The *clip* function maps an $\mathbb{R}_+^{n \times n}$ path matrix to a $\{0,1\}^{n \times n}$ matrix. The function is defined as

$$c : \mathbb{R}_+^{n \times n} \rightarrow \{0,1\}^{n \times n}$$

with a function rule of

$$c(\mathbf{Z})_{i,j} = \begin{cases} 1 & \text{if } \mathbf{Z}_{i,j} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

The general purpose of *clip* is to take a non-$\{0,1\}^{n \times n}$ path matrix and to "clip," or normalize, it to a $\{0,1\}^{n \times n}$ matrix. Thus, *clip* creates a filter that can then be applied to a composition to exclude paths.

If $\mathbf{A} \in \{0,1\}^{n \times n}$, then evident and useful properties are

- $c(\mathbf{A}) = \mathbf{A}$
- $c(n(\mathbf{A})) = n(c(\mathbf{A})) = n(\mathbf{A})$.

**Proposition 1.** *If* $\mathbf{Y}, \mathbf{Z} \in \mathbb{R}_+^{n \times n}$, *then*

$$c(\mathbf{Y} \circ \mathbf{Z}) = c(\mathbf{Y}) \circ c(\mathbf{Z}).$$

*Proof.* This property can be demonstrated given an entry-wise representation, where

$$c(\mathbf{Y}_{i,j} \cdot \mathbf{Z}_{i,j}) = c(\mathbf{Y}_{i,j}) \cdot c(\mathbf{Z}_{i,j}).$$

The equality holds for all cases where both entries are 0, both entries are greater than 0, and where one entry is 0 and the other is greater than 0. $\square$

**Proposition 2.** *If* $\mathbf{A}, \mathbf{B} \in \{0,1\}^{n \times n}$, *then*

$$n(\mathbf{A} \circ \mathbf{B}) = c\left(n(\mathbf{A}) + n(\mathbf{B})\right)$$

*Proof.* This proposition follows a similar pattern as De Morgan's law for boolean values, where $\neg(P \wedge Q) = \neg P \vee \neg Q$. However, because matrix addition over $\{0,1\}^{n \times n}$ matrices has the potential to yield a value of 2 if $n(\mathbf{A}_{i,j}) = n(\mathbf{B}_{i,j}) = 1$, *clip* will ensure that $c(n(\mathbf{A}) + c(\mathbf{B}))_{i,j} = 1$. $\square$

Likewise,

**Proposition 3.** *If* $\mathbf{A}, \mathbf{B} \in \{0,1\}^{n \times n}$, *then*

$$n(c(\mathbf{A} + \mathbf{B})) = n(\mathbf{A}) \circ n(\mathbf{B})$$

*Proof.* This proposition follows a similar pattern as De Morgan's law for boolean values, where $\neg(P \vee Q) = \neg P \wedge \neg Q$. $\qquad\square$

To use a scholarly example, it is possible to exclude all coauthorship and self-loop paths from a larger composite. For instance, if, as previously demonstrated,

$$\mathcal{A}^1 \cdot \mathcal{A}^{1^\top} \circ n(\mathbf{I})$$

defines a coauthorship path matrix and $\mathcal{A}^3$ denotes `cites` relations, then

$$\mathbf{Z} = \underbrace{\left(\mathcal{A}^1 \cdot \mathcal{A}^3 \cdot \mathcal{A}^{1^\top}\right)}_{\text{cites}} \circ \underbrace{n\left(c\left(\mathcal{A}^1 \cdot \mathcal{A}^{1^\top} \circ n(\mathbf{I})\right)\right)}_{\text{no coauthors}} \circ \underbrace{n(\mathbf{I})}_{\text{no self}}$$

is a `hasCited'` path matrix where citing one's coauthors and oneself is not considered a legal citation path. As previously demonstrated in §4.1, the first component (i.e. cites) generates a `hasCited` path matrix for all authors citing each other's articles, where coauthorship and self-citation are legal. The second component (i.e. no coauthors) applies the *not* function to a path matrix generated from a *clip* of a coauthorship path matrix. This excludes coauthors as being legal author citations. Finally, the third component (i.e. no self) disallows self-loops. The application of the two filter components removes paths that go from an author to his- or herself as well as to his or her respective coauthors.

With the help of the propositions and properties of the various operations of the path algebra, the above composition can be simplified. While the following simplification is lengthy, it utilizes many of the properties and propositions demonstrated hitherto. If $\mathbf{X} = \mathcal{A}^1 \cdot \mathcal{A}^3 \cdot \mathcal{A}^{1^\top}$ and $\mathbf{Y} = \mathcal{A}^1 \cdot \mathcal{A}^{1^\top}$, then

$$
\begin{array}{rll}
\mathbf{Z} & = \mathbf{X} \circ n(c(\mathbf{Y} \circ n(\mathbf{I}))) \circ n(\mathbf{I}) & \\
& = \mathbf{X} \circ n(c(\mathbf{Y}) \circ c(n(\mathbf{I})) \circ n(\mathbf{I}) & \text{prop. 1} \\
& = \mathbf{X} \circ n(c(\mathbf{Y}) \circ n(\mathbf{I})) \circ n(\mathbf{I}) & c(n(\mathbf{A})) = \mathbf{A} \\
& = \mathbf{X} \circ c(n(c(\mathbf{Y})) + n(n(\mathbf{I}))) \circ n(\mathbf{I}) & \text{prop. 2} \\
& = \mathbf{X} \circ c(n(c(\mathbf{Y})) + \mathbf{I}) \circ n(\mathbf{I}) & n(n(\mathbf{A})) = \mathbf{A} \\
& = \mathbf{X} \circ c(n(c(\mathbf{Y})) + \mathbf{I}) \circ c(n(\mathbf{I})) & c(n(\mathbf{A})) = n(\mathbf{A}) \\
& = \mathbf{X} \circ c(n(c(\mathbf{Y})) + \mathbf{I} \circ n(\mathbf{I}))) & \text{prop. 1} \\
& = \mathbf{X} \circ c(n(c(\mathbf{Y})) \circ n(\mathbf{I}) + \mathbf{I} \circ n(\mathbf{I}))) & \mathbf{A} \circ (\mathbf{B} + \mathbf{C}) = (\mathbf{A} \circ \mathbf{B}) + (\mathbf{A} \circ \mathbf{C}) \\
& = \mathbf{X} \circ c(n(c(\mathbf{Y})) \circ n(\mathbf{I})) & \mathbf{A} \circ n(\mathbf{A}) = \mathbf{0} \\
& = \mathbf{X} \circ c(n(c(\mathbf{Y}))) \circ c(n(\mathbf{I})) & \text{prop. 1} \\
& = \mathbf{X} \circ n(c(\mathbf{Y})) \circ n(\mathbf{I}) & c(n(\mathbf{A})) = n(\mathbf{A}).
\end{array}
$$

Thus,
$$\mathbf{Z} = \left(\mathcal{A}^1 \cdot \mathcal{A}^3 \cdot \mathcal{A}^{1\top}\right) \circ n \left(c\left(\mathcal{A}^1 \cdot \mathcal{A}^{1\top}\right)\right) \circ n(\mathbf{I}).$$
In words, only a single filter disallowing self-loops is necessary to yield the same result.

### 4.2.3. The Vertex Functions

In many cases, it is important to filter out particular paths from and to a vertex. Two useful functions are $v^-$ and $v^+$, where

$$v^- : \mathbb{R}_+^{n \times n} \times \mathbb{N} \to \{0, 1\}^{n \times n},$$

$$v^-(\mathbf{Z}, p)_{i,j} = \begin{cases} 1 & \text{if } \sum_{l \in V} \mathbf{Z}_{i,l} > p \\ 0 & \text{otherwise} \end{cases}$$

turns a complete row into an all 1-row if the sum of row entries is greater than $p$ and

$$v^+ : \mathbb{R}_+^{n \times n} \times \mathbb{N} \to \{0, 1\}^{n \times n},$$

$$v^+(\mathbf{Z}, p)_{i,j} = \begin{cases} 1 & \text{if } \sum_{l \in V} \mathbf{Z}_{l,j} > p \\ 0 & \text{otherwise} \end{cases}$$

turns a complete column into an all 1-column if the sum of the column entries is greater than $p$. The function $v^-$ is used to select paths outgoing from particular vertices and $v^+$ is used to select paths incoming to particular vertices. Moreover, by providing $p$, it excludes those vertices with less than $p$ paths outgoing from or incoming to it. For the sake of brevity, when no $p$ is supplied, it is assumed that $p = 0$.

Some useful properties of the vertex filter are

- $v^-(\mathbf{R}_i) = \mathbf{R}_i$
- $v^+(\mathbf{C}_i) = \mathbf{C}_i$
- $v^+(\mathbf{E}_{i,j}) \circ v^-(\mathbf{E}_{i,j}) = \mathbf{E}_{i,j}$
- $v^-(\mathbf{Z} \circ \mathbf{R}_i) = v^-(\mathbf{Z}) \circ \mathbf{R}_i$
- $v^+(\mathbf{Z} \circ \mathbf{C}_i) = v^+(\mathbf{Z}) \circ \mathbf{C}_i$
- $v^-(\mathbf{Z}, p) = v^+(\mathbf{Z}^\top, p)^\top$
- $v^+(\mathbf{Z}, p) = v^-(\mathbf{Z}^\top, p)^\top.$

13

To demonstrate the use of the *vertex* function, consider the multi-relational tensor $\mathcal{A}$ that includes journals, articles, and subject categories, where $\mathcal{A}^3$ denotes `cites` $: A \rightarrow A$, $\mathcal{A}^4$ denotes `contains` $: J \rightarrow A$, $\mathcal{A}^5$ denotes `category` $: J \rightarrow S$, and vertex 1 denotes the subject category "social science." A social science journal citation matrix can be created from $\mathcal{A}$ in which a path exists between journals if and only if an article contained in a journal cites an article contained in another journal. Furthermore, only those citing and cited articles are considered that are in social science journals. Thus, the social science journal citation path matrix is defined as

$$\mathbf{Z} = \underbrace{\left[ v^- \left( \mathbf{C}_1 \circ \mathcal{A}^5 \right) \circ \mathcal{A}^4 \right]}_{\text{soc.sci. journal articles}} \cdot \mathcal{A}^3 \cdot \underbrace{\left[ \mathcal{A}^{4\top} \circ v^+ \left( \mathbf{R}_1 \circ \mathcal{A}^{5\top} \right) \right]}_{\text{articles in soc.sci. journals}}.$$

First, a *vertex*-created filter is applied to remove all articles that are not contained in social science journals. Next, the articles that these articles cite is determined. Finally, those articles not in social science journals are filtered out using another *vertex*-created filter. Thus, a citation path matrix is generated which only includes social science journals.

Using the various aforementioned *vertex* function properties, the above expression for a social science journal citation path matrix can be simplified, because

$$
\begin{aligned}
v^+ \left( \mathbf{R}_1 \circ \mathcal{A}^{5\top} \right) &= v^+ \left( (\mathbf{C}_1 \circ \mathcal{A}^5)^\top \right) & & \mathbf{R}_1 = \mathbf{C}_1^\top \\
&= v^- \left( \mathbf{C}_1 \circ \mathcal{A}^5 \right)^\top & & v^-(\mathbf{Z}) = \\
& & & \quad v^+(\mathbf{Z}^\top)^\top.
\end{aligned}
$$

Therefore, given the above and because $\mathbf{A}^\top \circ \mathbf{B}^\top = (\mathbf{A} \circ \mathbf{B})^\top$,

$$\mathbf{Z} = \left[ v^- \left( \mathbf{C}_1 \circ \mathcal{A}^5 \right) \circ \mathcal{A}^4 \right] \cdot \mathcal{A}^3 \cdot \left[ v^- \left( \mathbf{C}_1 \circ \mathcal{A}^5 \right) \circ \mathcal{A}^4 \right]^\top.$$

The above composition reuses the computation for determining which articles are contained in social science journals by simply reversing the directionality of the final component. The two bracketed components represent `contains` such that the domain is social science journals and the range is articles. The ability to algebraically manipulate path expressions is one of the primary benefits of utilizing an algebraic structure to map a multi-relational network to a single-relational network.

### 4.3. The Weight Operation

Composed paths can be weighted using ordinary matrix scalar multiplication. Given a scalar value of $\lambda \in \mathbb{R}$, $\lambda \mathbf{Z}$ will weight all the paths in $\mathbf{Z}$ by $\lambda$. This operation is useful when merging path matrices in such a way as to make one path matrix more or less significant than another path matrix. The next subsection, §4.4, discusses the operation of merging two path matrices and presents an example that includes the weight operation.

### 4.4. The Merge Operation

Ordinary matrix addition can be used to merge two path matrices. For example, consider the multi-relational tensor $\mathcal{A}$, where $\mathcal{A}^1$ denotes `authored` and $\mathcal{A}^6$ denotes `developed`, `authored` maps humans to articles, and `developed` maps humans to programs. Furthermore, consider a definition of collaboration that includes both the coauthorship of articles and the co-development of software, where article coauthorship is weighted as being slightly more important than co-development. The path matrix

$$\mathbf{Z} = 0.6 \underbrace{\left( \mathcal{A}^1 \cdot \mathcal{A}^{1^\top} \circ n(\mathbf{I}) \right)}_{\text{coauthorship}} + 0.4 \underbrace{\left( \mathcal{A}^6 \cdot \mathcal{A}^{6^\top} \circ n(\mathbf{I}) \right)}_{\text{co-development}}$$

merges the article and software program collaboration path matrices as specified by their respective weights of 0.6 and 0.4. The resultant path matrix denotes article and software program collaboration. Finally, using the properties and propositions of the path algebra, a simplification of the previous composition is

$$\mathbf{Z} = \left[ 0.6 \left( \mathcal{A}^1 \cdot \mathcal{A}^{1^\top} \right) + 0.4 \left( \mathcal{A}^6 \cdot \mathcal{A}^{6^\top} \right) \right] \circ n(\mathbf{I}).$$

## 5. Network Analysis Applications

The previous sections presented various elements and operations of the path algebra that can be applied to a multi-relational tensor in $\{0, 1\}^{n \times n \times m}$ in order to derive a "semantically-rich" single-relational path matrix in $\mathbb{R}_+^{n \times n}$. The resultant path matrix yields the number of paths from vertex $i$ to vertex $j$ as determined by the operations performed. The path matrix can be considered a weighted single-relational network of the form $G = (V, E, w)$, where $w : E \to R_+$. Many single-relational network analysis algorithms require either a $\{0, 1\}^{n \times n}$ matrix or a $[0, 1]^{n \times n}$ weighted or stochastic matrix. The

resultant path matrix can be manipulated in various ways (e.g. normalized out going weight distribution) to yield a matrix that can be appropriately used with the known single-relational network analysis algorithms. This section discusses the connection of a path matrix to a few of the more popular single-relational network analysis algorithms. However, before discussing the single-relational network analysis algorithms, the next subsection discusses the relationship between the path algebra and multi-relational graph query languages.

*5.1. Relationship to Multi-Relational Graph Query Languages*

The presented path algebra has many similarities to graph query languages such as GraphLog [26], Path Query Language (PQL) [27], and SPARQL Protocol and RDF Query Language (SPARQL) [28]. All of these languages serve a similar function of querying a graph for data, though they have different levels of expressivity (e.g. some have mechanisms to include extension functions in the query, to perform regular expressions on the vertex names, and the ability to perform recursion). However, what these languages have in common is the ability to perform graph pattern matching. Graph pattern matching is explained using an example. Suppose the multi-relational network representation $H \subseteq (V \times \Omega \times V)$, where $V$ is the set of vertices and $\Omega$ is the set of edges labels. Given

$$Z = \{?x \mid (?x, \texttt{authoredBy}, \texttt{marko}) \in H\},$$

$Z$ is the set of all articles authored by Marko. In short, ?-variables are used to bind to particular vertices and must hold for the duration of the query. This is made more salient in the following, more complicated example:

$$\begin{aligned}
Z = \{?y \mid &(?x, \texttt{authoredBy}, \texttt{marko}) \in H \\
&\wedge (?x, \texttt{cites}, ?y) \in H \\
&\wedge (\texttt{JOI}, \texttt{contains}, ?y) \in H \\
&\wedge ?x \neq ?y \}.
\end{aligned}$$

In this example, the set $Z$ is the set of all articles that are

1. contained in the Journal of Informetrics (JOI),
2. cited by Marko's authored articles, and
3. are not articles authored by Marko.

In SPARQL, this is represented as

```
SELECT ?y WHERE {
  ?x authoredBy marko .
  ?x cites ?y .
  JOI contains ?y .
  FILTER (?x != ?y) }.
```

In the presented path algebra, this same query is represented as

$$\mathbf{Z} = c \left( \underbrace{\left[ \left( \mathbf{C}_2 \circ \mathcal{A}^{1\top} \right) \cdot \mathcal{A}^1 \circ \mathbf{I} \right]}_{\text{marko's articles}} \cdot \underbrace{\left[ \mathcal{A}^3 \circ n \left( v^- \left( \mathbf{C}_2 \circ \mathcal{A}^{1\top} \right)^{\top} \right) \right]}_{\text{citations to non-marko articles}} \cdot \underbrace{\left[ \mathbf{C}_3 \circ \mathcal{A}^{4\top} \right]}_{\text{contained in joi}} \right),$$

where vertex 2 is Marko and vertex 3 is the Journal of Informetrics. The resulting $\mathbf{Z}$ path matrix is a {0,1}-matrix, where the row vertices that have a 1 as an entry are equivalent to those vertices that bound to ?$y$ in the related SPARQL query. While the algebraic form may be considered more cumbersome than the SPARQL form, the benefit of using the path algebra is that the query can be simplified through algebraic manipulations, and that it has a convenient implementation using existing linear algebra toolkits. Moreover, a hybrid approach can be utilized that leverages the syntactic convenience of the standard graph query languages and the algebraic properties of the path algebra. That is, as many of the queries used in graph query languages can be specified in the path algebra, it is possible to optimize such queries in the algebra and then convert them back to the query language for execution.

Finally, the path algebra provides more information than a binding of variables to vertices. For instance, in the previous example, without the *clip* function, the resultant $\mathbf{Z}$ would return how many paths exist from Marko's articles to those non-Marko cited articles that are contained in the Journal of Informetrics. In this way, some Journal of Informetrics articles may be deemed more "appropriate" to the query (as there may be more paths to them). The result path matrix can be seen as a weighted single-relational network that can be manipulated further by single-relational network analysis algorithms or used in a larger path expression.

*5.2. Shortest Path Calculation*

The family of geodesic algorithms are based on the calculation of the shortest path between vertices [1]. Example shortest path metrics include:

17

- *eccentricity*: defined for a vertex as the longest shortest path to all other vertices [6],

- *radius*: defined for the network as the smallest eccentricity value for all vertices,

- *diameter*: defined for the network as the largest eccentricity value for all vertices,

- *closeness*: defined for a vertex as the mean shortest path of a vertex to all other vertices [3],

- *betweenness*: defined for a vertex as the number of shortest paths that a vertex is a part of [5].

A straightforward (though computationally expensive) way to calculate the shortest path between any two vertices $i$ and $j$ in an adjacency matrix $\mathbf{A}$ is to continue to raise the adjacency matrix by a power until $\mathbf{A}_{i,j}^{(t)} > 0$. The first $t$ where $\mathbf{A}_{i,j}^{(t)} > 0$ denotes the length of the shortest path between vertex $i$ and $j$ as $t$. This same principle holds for calculating the shortest path in a path matrix $\mathbf{Z}$ and thus, the resulting path matrix generated from a path composition can be used to determine "semantically-rich" shortest paths between vertices.

*5.3. Diffusing an Energy Vector through $\mathcal{A}$*

Many network analysis algorithms can be represented as an energy diffusion, where

$$f : \mathbb{R}_+^n \times \mathbb{R}_+^{n \times n} \to \mathbb{R}_+^n$$

maps a row-vector of size $n$ (i.e. "energy vector") and a matrix of size $n \times n$ to a resultant energy vector of size $n$. Algorithms of this form include eigenvector centrality [7], PageRank [8], and spreading activation [29, 30] to name but a few.

With respect to eigenvector centrality and PageRank, the general form of the algorithm can be represented as a problem in finding the primary eigenvector of an adjacency matrix such that $\pi \mathbf{A} = \lambda \pi$, where $\pi \in \mathbb{R}^n$ and $\pi$ is the energy vector being diffused. Both algorithms can be solved using the "power method." In one form of the power method, the solution is found by iteratively multiplying $\pi$ by $\mathbf{A}$ until $\pi$ converges to a stable set of values as defined by $||\pi^{(t-1)} - \pi^{(t)}||_2 < \epsilon$ for some small $\epsilon \in \mathbb{R}_+$. In another form, the problem is framed as finding which $t$-power of $\mathbf{A}$ will yield a $\pi$ such that

$\pi \mathbf{A}^{(t)} = \lambda \pi$. With respect to spreading activation, the same power method can be used; however, the purpose of the algorithm is not to find the primary eigenvector, but instead to propagate an energy vector some finite number of steps. Moreover, the total energy flow through each vertex at the end of a spreading activation algorithm is usually what is determined.

### 5.3.1. The PageRank Path Matrix

The PageRank algorithm was developed to determine the centrality of web pages in a web citation network [8] and since, has been used as a general network centrality algorithm for various other types of networks including bibliographic [31, 32], social [33], and word networks [34]. The web citation network can be defined as $G = (V, E)$, where $V$ is a set of web pages and $E \subseteq (V \times V)$ is the set of directed citations between web pages (i.e. `href`). The interesting aspect of the PageRank algorithm is that it "distorts" the original network $G$ by overlaying a "teleportation" network in which every vertex is connected to every other vertex by some weight defined by $\delta \in (0, 1]$. The inclusion of the teleportation network ensures that the resulting hybrid network is strongly connected[7] and thus, the resultant primary eigenvector of the network is a positive real-valued vector.

In the matrix form of PageRank, there exist two adjacency matrices in $[0, 1]^{n \times n}$ denoted

$$\mathcal{P}^1_{i,j} = \begin{cases} \frac{1}{\Gamma(i)} & \text{if } (i, j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

and

$$\mathcal{P}^2_{i,j} = \frac{1}{|V|},$$

where $\Gamma(i)$ is the out degree of vertex $i$. $\mathcal{P}^1$ is a row-stochastic adjacency matrix and $\mathcal{P}^2$ is a fully connected adjacency matrix known as the teleportation matrix. The purpose of PageRank is to identify the primary eigenvector of a merged, weighted path matrix of the form

$$\mathbf{Z} = \delta \mathcal{P}^1 + (1 - \delta)\mathcal{P}^2.$$

$\mathbf{Z}$ is guaranteed to be a strongly connected single-relational path matrix

---

[7]Strongly connected means that there exists a path from every vertex to every other vertex. In the language of Markov chains, the network is irreducible and recurrent.

because there is some probability (defined by $1 - \delta$) that every vertex is reachable by every other vertex.

### 5.3.2. Constrained Spreading Activation

The concept of spreading activation was made popular by cognitive scientists and the connectionist approach to artificial intelligence [29, 35, 36], where the application scenario involves diffusing an energy vector through an artificial neural network in order to simulate the neural process of a spreading activation potential in the brain. Spreading activation can be computed in a manner analogous to the power method of determining the primary eigenvector of an adjacency matrix. However, spreading activation does not attempt to find a stationary energy distribution in the network and moreover, usually includes a decay function or activation/threshold function that can yield a resultant energy vector whose sum is different than the initial energy vector.

The neural network models of connectionism deal with weighted, single-relational networks. Spreading activation was later generalized to support multi-relational networks [30]. While there are many forms of spreading activation on single-relational networks and likewise, many forms on multi-relational networks, in general, a spreading activation algorithm on a multi-relational network is called constrained spreading activation as not all edges to and from a vertex are respected equally. Constrained spreading activation has been used for information retrieval on the web [30, 37, 38], semantic vertex ranking [18, 39], and collaborative filtering [40]. The path algebra provides a means by which to separate the spreading activation algorithm from the data structure being computed on. That is, the *constrained* aspect of the algorithm is defined by the path algebra and the *spreading activation* aspect of the algorithm is defined by standard, single-relational spreading activation.

### 5.4. Mixing Patterns

Given a network and a scalar or categorical property value for each vertex in the network, it is possible to determine whether the network is assortative or disassortative with respect to that property [41]. Example scalar properties for journal vertices in a scholarly network include impact factor ranking, years in press, cost for subscription, etc. Example categorical properties include subject category, editor in chief, publisher, etc. A network is assortative with respect to a particular vertex property when vertices of like

property values tend to be connected to one another. Colloquially, assortativity can be understood by the phrase "birds of a feather flock together," where the "feather" is the property binding individuals [42]. On the other hand, a network is disassortative with respect to a particular vertex property when vertices of unlike property values tend to be connected to one another. Colloquially, disassortativity can be understood by the phrase "opposites attract."

There are two primary components needed when calculating the assortativity of a network. The first component is the network and the second component is a set of property values for each vertex in the network. A popular publication defining the assortative mixing for scalar properties uses the parametric Pearson correlation of two vectors [43].[8] One vector is the scalar value of the vertex property for the vertices on the tail of all edges. The other vector is the scalar value of the vertex property for the vertices on the head of all the edges. Thus, the length of both vectors is $|E|$ (i.e. the total number of edges in the single-relational network). Formally, the correlation is defined as

$$r = \frac{|E| \sum_i j_i k_i - \sum_i j_i \sum_i k_i}{\sqrt{\left[|E| \sum_i j_i^2 - \left(\sum_i j_i\right)^2\right]\left[|E| \sum_i k_i^2 - \left(\sum_i k_i\right)^2\right]}},$$

where $j_i$ is the scalar value of the vertex on the tail of edge $i$, and $k_i$ is the scalar value of the vertex on the head of edge $i$. The correlation coefficient $r$ is in $[-1, 1]$, where $-1$ represents a fully disassortative network, 0 represents an uncorrelated network, and 1 represents a fully assortative network. On the other hand, for categorical properties, the equation

$$r = \frac{\sum_a e_{aa} - \sum_a i_a j_a}{1 - \sum_a i_a j_a}$$

yields a value in $[-1, 1]$ as well, where $e_{aa}$ is the number of edges in the network that have property value $a$ on both ends, $i_a$ is the number of edges in the network that have property value $a$ on their tail vertex, and $j_a$ is the number of edges that have property value $a$ on their head vertex [41].

---

[8]Note that scalar value distributions may not be normally distributed and thus, in such cases, a non-parametric correlation such as the Spearman $\rho$ may be the more useful correlation coefficient.

Given a path matrix, assortativity is calculated on paths, not on edges, where there may be many paths (possibly weighted) between any two vertices.[9] Thus, a weighted correlation is required [44]. Let $j_i$ denote the scalar property value of the vertex on the tail of path $i$ and $k_i$ denote the scalar property value of the vertex on the head of path $i$. The previous scalar assortativity equation can be generalized such that if $z_i$ is the fraction of path weight in $\mathbf{Z}$ for path $i$, then

$$r = \frac{\text{cov}_{j,k}}{\sqrt{\text{cov}_{j,j}\text{cov}_{k,k}}},$$

where

$$\text{cov}_{j,k} = \frac{1}{\sum_i z_i}\left[\sum_i z_i\left(j_i - \frac{\sum_i z_i j_i}{\sum_i z_i}\right)\left(k_i - \frac{\sum_i z_i k_i}{\sum_i z_i}\right)\right].$$

Similarly, for categorical vertex properties,

$$r = \frac{\sum_a e_{aa} - \sum_a i_a j_a}{1 - \sum_a i_a j_a}$$

where $e_{aa}$ is the total path weight of paths that have tail and head vertices with a property value of $a$, $i_a$ is the total path weight of all paths that have a tail vertex with a property value of $a$, and $j_a$ is the total path weight of all paths that have a head vertex with a property value of $a$.

## 6. Conclusion

The number of algorithms and toolkits for single-relational networks far exceeds those currently available for multi-relational networks. However, with the rising popularity of the multi-relational network, as made evident by the Web of Data initiative and the multi-relational RDF data structure [45], there is a need for methods that port the known single-relational network analysis algorithms over to these multi-relational domains. A convenient

---

[9]It is important to note that with multi-relational networks, vertex property values can be encoded in the network itself. For instance, given the scholarly network example of previous, the subject category "social science" is a vertex adjacent to a set of journal vertices according to the relation `category` : $J \to S$. More generally, vertex property values may be determined through path composition.

method to do so is a path algebra. Path algebras have been used extensively for analyzing paths in single-relational networks [46, 47] and the application to multi-relational networks can prove useful. The multi-relational path algebra presented in this article operates on an $n \times n \times m$ tensor representation of a multi-relational network. By means of a series of operations on two-way "slices" of this tensor, a "semantically-rich" $n \times n$ single-relational path matrix can be derived. The resulting path matrix represents a single-relational network. This single-relational network may then be subjected to any of the known single-relational network analysis algorithms. Thus, the presented path algebra can be used to expose multi-relational networks to single-relational network analysis algorithms.

# References

[1] S. Wasserman, K. Faust, Social Network Analysis: Methods and Applications, Cambridge University Press, Cambridge, UK, 1994.

[2] U. Brandes, T. Erlebach (Eds.), Network Analysis: Methodolgical Foundations, Springer, Berling, DE, 2005.

[3] A. Bavelas, Communication patterns in task oriented groups, The Journal of the Acoustical Society of America 22 (1950) 271–282.

[4] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271.

[5] L. C. Freeman, A set of measures of centrality based on betweenness, Sociometry 40 (35–41).

[6] F. Harary, P. Hage, Eccentricity and centrality in networks, Social Networks 17 (1995) 57–63.

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

[8] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Computer Networks and ISDN Systems 30 (1–7) (1998) 107–117.

[9] F. R. K. Chung, Spectral Graph Theory, American Mathematical Society, 1997.

[10] M. E. J. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical Review E 69 (2004) 026113.
URL http://arxiv.org/abs/cond-mat/0308217

[11] M. Girvan, M. E. J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences 99 (2002) 7821–7826.
URL http://arxiv.org/abs/cond-mat/0112110

[12] M. E. J. Newman, Finding community structure in networks using the eigenvectors of matrices, Physical Review E 74. arXiv:physics/0605087.
URL http://arxiv.org/abs/physics/0605087

[13] J. F. Sowa, Principles of Semantic Networks: Explorations in the Representation of Knowledge, Morgan Kaufmann, San Mateo, CA, 1991.

[14] M. A. Rodriguez, A multi-relational network to support the scholarly communication process, International Journal of Public Information Systems 2007 (1) (2007) 13–29.
URL http://arxiv.org/abs/cs/0601121

[15] J. Bollen, M. A. Rodriguez, H. Van de Sompel, L. L. Balakireva, A. Hagberg, The largest scholarly semantic network...ever., in: Proceedings of the World Wide Web Conference, ACM Press, New York, NY, 2007, pp. 1247–1248. doi:10.1145/1242572.1242789.

[16] T. Berners-Lee, J. A. Hendler, Publishing on the Semantic Web, Nature 410 (6832) (2001) 1023–1024. doi:10.1038/35074206.

[17] M. A. Rodriguez, Data Management in the Semantic Web, Nova Publishing, 2009, Ch. Interpretations of the Web of Data.
URL http://arxiv.org/abs/0905.3378

[18] M. A. Rodriguez, Grammar-based random walkers in semantic networks, Knowledge-Based Systems 21 (7) (2008) 727–739. doi:10.1016/j.knosys.2008.03.030.
URL http://arxiv.org/abs/0803.4355

[19] S. Lin, Interesting instance discovery in multi-relational data, in: D. L. McGuinness, G. Ferguson (Eds.), Proceedings of the Conference on Innovative Applications of Artificial Intelligence, MIT Press, 2004, pp. 991–992.

[20] D. Cai, Z. Shao, X. He, X. Yan, J. Han, Community mining from multi-relational networks, in: Proceedings of the 2005 European Conference on Principles and Practice of Knowledge Discovery in Databases, Vol. 3721 of Lecture Notes in Computer Science, Springer, Porto, Portugal, 2005, pp. 445–452. doi:10.1007/11564126.

[21] H. Zhuge, L. Zheng, Ranking semantic-linked network, in: Proceedings of the International World Wide Web Conference, Budapest, Hungary, 2003.

[22] B. Aleman-Meza, C. Halaschek-Wiener, I. B. Arpinar, C. Ramakrishnan, A. P. Sheth, Ranking complex relationships on the semantic web, IEEE Internet Computing 9 (3) (2005) 37–44. doi:http://doi.ieeecomputersociety.org/10.1109/MIC.2005.63.

[23] T. G. Kolda, B. W. Bader, J. P. Kenny, Higher-order web link analysis using multilinear algebra, in: Proceedings of the International Conference on Data Mining ICDM'05, IEEE, 2005, pp. 242–249.

[24] G. Chartrand, Introductory Graph Theory, Dover, 1977.

[25] R. Horn, C. Johnson, Topics in Matrix Analysis, Cambridge University Press, 1994.

[26] M. P. Consens, A. O. Mendelzon, Graphlog: a visual formalism for real life recursion, in: Proceedings of the Symposium on Principles of Database Systems, ACM, New York, NY, 1990, pp. 404–416. doi:10.1145/298514.298591.

[27] U. Leser, A query language for biological networks, Bioinformatics 21 (2) (2005) 33–39. doi:10.1093/bioinformatics/bti1105.

[28] E. Prud'hommeaux, A. Seaborne, SPARQL query language for RDF, Tech. rep., World Wide Web Consortium (October 2004).
URL http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/

[29] J. R. Anderson, A spreading activation theory of memory, Journal of Verbal Learning and Verbal Behaviour 22 (1983) 261–295.

[30] P. R. Cohen, R. Kjeldsen, Information retrieval by constrained spreading activation in semantic networks, Information Processing and Management 23 (4) (1987) 255–268.

[31] J. Bollen, M. A. Rodriguez, H. Van de Sompel, Journal status, Sciento-metrics 69 (3) (2006) 669–687. doi:10.1007/s11192-006-0176-z.
URL http://arxiv.org/abs/cs/0601030

[32] P. Chen, H. Xie, S. Maslov, S. Redner, Finding scientific gems with Google, Journal of Informetrics 1 (1) (2007) 8–15.
URL http://arxiv.org/abs/physics/0604130

[33] X. Liu, J. Bollen, M. L. Nelson, H. Van de Sompel, Co-authorship net-works in the digital library research community, Information Processing and Management 41 (6) (2006) 1462–1480.
URL http://arxiv.org/abs/cs.DL/0502056

[34] R. Mihalcea, P. Tarau, E. Figa, Pagerank on semantic networks, with application to word sense disambiguation, in: Proceedings of the International Conference on Computational Linguistics, Associa-tion for Computational Linguistics, Morristown, NJ, 2004, p. 1126. doi:10.3115/1220355.1220517.

[35] A. M. Collins, E. F. Loftus, A spreading activation theory of semantic processing, Psychological Review 82 (1975) 407–428.

[36] D. E. Rumelhart, J. L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, 1993.

[37] F. Crestani, Application of spreading activation techniques in informa-tion retrieval, Artificial Intelligence Review 11 (6) (1997) 453–582.

[38] F. Crestani, P. L. Lee, Searching the web by constrained spreading acti-vation, Information Processing and Management 36 (4) (2000) 585–605.

[39] M. A. Rodriguez, Social decision making with multi-relational networks and grammar-based particle swarms, in: Proceedings of the Hawaii In-ternational Conference on Systems Science, IEEE Computer Society, Waikoloa, Hawaii, 2007, pp. 39–49. doi:10.1109/HICSS.2007.487.
URL http://arxiv.org/abs/cs.CY/0609034

[40] J. Griffith, C. O'Riordan, H. Sorensen, Knowledge-Based Intelligent Information and Engineering Systems, Vol. 4253 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 2006, Ch. A Constrained

Spreading Activation Approach to Collaborative Filtering, pp. 766–773. doi:10.1007/11893011.

[41] M. E. J. Newman, Mixing patterns in networks, Physical Review E 67 (2) (2003) 026126. doi:10.1103/PhysRevE.67.026126.
URL http://arxiv.org/abs/cond-mat/0209450

[42] M. McPherson, L. Smith-Lovin, J. Cook, Birds of a feather: Homophily in social networks, Annual Review of Sociology 27 (2001) 415–444.

[43] M. E. J. Newman, Assortative mixing in networks, Physical Review Letters 89 (20) (2002) 208701.
URL http://arxiv.org/abs/cond-mat/0205405/

[44] J. M. Bland, D. G. Altman, Calculating correlation coefficients with repeated observations: Part 2–correlation between subjects, British Medical Journal 310 (6980) (1995) 633.

[45] E. Miller, An introduction to the Resource Description Framework, Bulletin of the American Society for Information Science and Technology 25 (1) (1998) 15–19. doi:10.1002/bult.105.

[46] B. Carre, Graphs and Networks, Oxford University Press, 1979.

[47] R. Manger, A new path algebra for finding paths in graphs, in: Proceedings of the International Conference on Information Technology Interfaces, Vol. 1, 2004, pp. 657–662. doi:10.1109/ITI.2004.242700.