Open Risk White Paper

Connecting the Dots: Towards a Graph Calculus for Decentralized Social Networks

Authors: Philippos Papadopoulos

September 1, 2026

www.openriskmanagement.com

The open future of risk management

Contents

SUMMARY

In this White Paper we develop graph representations of decentralized (heterogeneous) Social Web networks. We build primarily on the machinery of multilayer networks with their associated tensor representations. We expand on the specialized subclass of so-called layer-disjoint networks as they offer powerful tools for modeling complex online interactions of Actors within distinct Communities. We focus on the quantitative metrics and tools that can be expressed using such a framework, starting with basic building blocks and establishing the practical correspondence between adjacency tensors and supra-adjacency matrices, culminating with a discussion of the construction and use of tensor graph Laplacians. These tools can on the one hand help ex-post empirical (statistical) work in analyzing the characteristics of current decentralized networks, on the other hand they enable ex-ante simulation and quantification of heterogeneous network properties and behavior.

Further Resources

- The Open Risk Academy offers a range of online courses around risk and sustainable portfolio management, which utilize the latest in interactive eLearning tools.
- The Open Risk Manual is an open online repository of information for risk management developed and maintained by Open Risk.
- The Open Source Repository contains a variety of open source tools and resources.
- More content: Open Risk White Papers and the Open Risk Blog

About Open Risk

Open Risk is an independent provider of training and risk analysis tools to the broader financial services community. Our mission is captured by the motto: The open future of risk management. Learn more about our mission and projects at: www.openriskmanagement.com

1 Motivation and Objectives

In recent times significant developments in both the availability and capabilities of online digital networks operating over Internet infrastructure have enabled unprecedented connectivity between individuals. This has led over time to the formation of significant and diverse online communities. Of particular importance is the emergence of large-scale *Social Web* networks, where many individuals and organizations engage and interact in what are essentially public fora. These digital infrastructures compete with, and have even surpassed, traditional communication channels and patterns.

There are unprecedented and still unfolding impacts from such networks on social, political and economic structure. This has prompted a global discussion and questioning of how network topologies, algorithmic choices and other design aspects determine control, and augment or diminish the agency of participating individuals. One of the most important considerations is the degree of *centralization* of important functions of the network. The currently prevalent Social Web platforms are entirely centralized: Actual technical implementations may be using Web primitives to some degree (thus nominally are part of the global Web), but users connect via *client software*, to centrally controlled collections of *servers*. The so-called *social graph* (who "connects" with whom), and all important functions of the social network (including authentication, moderation, prioritization of content etc.) are entirely contained within and controlled by one entity. This is a rather unusual state of affairs, not reflecting either how information exchange happens outside Internet technologies, nor any inherent limitations of digital networks in supporting decentralization. In fact as stated by the very inventor of the Web, Sir Tim Berners-Lee, *the Web is already decentralized*.

1.1 The challenge and promise of decentralized, heterogeneous networks

Not surprising then, that there are nascent recent developments in online network technologies which use both existing and new Internet / Web protocols to establish novel *decentralized* communication patterns [1, 2]. The general design principle of such decentralized frameworks is to form an *ensemble* of distinct network elements, which communicate with each other via well defined protocols - even while remaining independently controlled platforms. While the potentially transformative impact of so-called **Decentralized Online Social Networks** (DOSN) is increasingly recognized [3], adoption (as of 2026) of such platforms lags by several orders of magnitude the centralized platforms. As Staschen et al. state:

> *Open, decentralized social networks built on standardized protocols still operate largely at the margins. Perceptions of limited reach or complex usability continue to hinder their broader adoption. The current state of decentralized platforms should therefore not be seen as a final outcome but as a transitional phase - one shaped by volunteer-driven initiatives, scarce resources and competition with globally dominant platforms.*

Bridging the gap from the current adoption of DOSN to a Web scale reality will involve significant further technical and socioeconomic developments [3]: Decentralized alternatives must become more user-friendly, appealing and accessible; Clear legal, institutional and financial structures are required to enable stable governance and fair competition; Open networks must be recognized as public-interest digital infrastructure and supported politically, civically and culturally.

While there are clearly significant and diverse challenges towards that goal, there are unique advantages to decentralization that are hiding in plain sight. The universality of such networks, the ability to connect

very diverse populations and activities is one such intrinsic strength. Centralized platforms up to a point successfully *emulated* such universality, and this has been indeed a key point of their appeal, but at high and unsustainable societal cost. A successful decentralized end-stage will undoubtedly deserve the name **Web 2.0**, as it will enable bidirectional exchanges and federation between independent entities over the entire Web.

A conceptual analysis tool that can facilitate the study and maybe influence the maturing of such new organizational patterns is so-called *social network analysis*. This is the discipline of analyzing social network structure and behavior on the basis of quantitative information. There is already effort among both academics and practitioners (developers and operators of online communities) that analyzes various Social Web phenomena using *graph theory* and *network analysis*, see e.g., [4, 5, 6, 7, 8, 9]

Performing social network analysis of decentralized Social Web networks, motivates the development of suitable *mathematical representations*. In a previous Open Risk White Paper in the *Connecting the Dots* series [10] we explored a *graph representation* of online communication networks that are organized according to the *ActivityPub* protocol [1]. We discussed the main relevant features of that protocol, and the broader application ecosystem around it that shapes emerging online network topologies. We developed a stylized description of ActivityPub compliant networks as a *mathematical multilayer network* and an associated tensor representation of adjacency.

Here we continue this work, focusing on enriching and developing the framework further, towards quantitative tools that can be built on top of it. The broader toolkit that we will tap into is going under a variety of names, such as *multilayer networks*, *multiplex networks* or *multidimensional graphs* [11, 12, 13].

1.2 Designing a suitable mathematical network representation

A large variety of Social Web network types can be represented using the general multilayer-network framework. Multilayer networks or graphs are flexible enough to enable formulating very general classes of computational problems. They rely on intuitive primitives such as *graph nodes and edges*, and, importantly, provide associated linear algebra representations that enable concise descriptions of the network and quantitative analyses. Mathematical frameworks for multilayer networks that use tensor algebra have been presented in [13, 12, 14] and we will follow here their general notation and definitions, but adapting them to the subclass of networks that is our focus. While staying within the broader multilayer network toolkit, the abstractions we will use entail choices which are guided by specific analysis objectives. We condition on the fact that in Web-scale decentralized networks participating nodes will (in general) not have identical roles and functions in the overall communication flow. A preview of this diversity is given by the list of distinct server software that implements the ActivityPub protocol, which is already extensive [15], even if currently the majority are instances of Mastodon [16]. Even within the relatively more homogeneous ensemble of Mastodon instances, there is significant observed variation of policies and configurations.

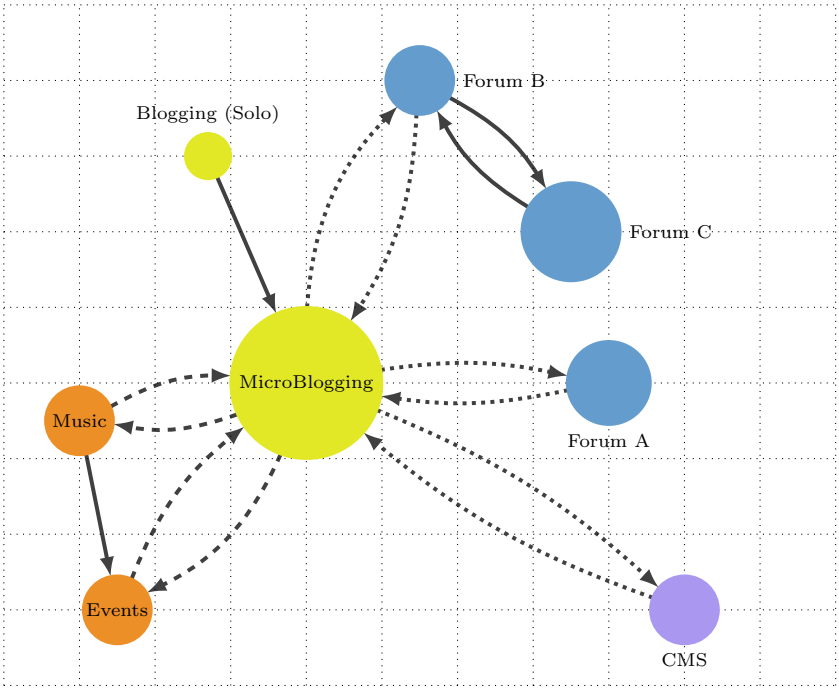
Reflecting and accommodating in the network model the presence of different types of *Communities*, leads to considering models of *heterogeneous* Social Web networks. Our archetype Social Web network model will be loosely based on the ActivityPub protocol (from where the Actor nomenclature derives) and we will develop and explore a suitably defined Actor/Community graph. At present the Community dimension of Activipub networks (the so-called Fediverse) is not fully developed, but it can already be proxied by, e.g., treating different Mastodon instances / servers as distinct Communities. The lack of

developed heterogeneity is even more visible in the case of the alternative ATProto networks [2], which at this moment retain very pronounced centralization features.

The central concept in our canonical network model are **Actors**. Loosely speaking, Actors pursue online Activities [17, 18] and facilitating these activities is the reason the network exists. Actors are naturally mapped to being nodes of a graph. Another central concept is that of the *Social Graph*. Without going yet into formal definitions we can naturally map Following/Followed type activities into a graph, but "negative follows" such as blocking activities and many other activities ("likes/dislikes") can also be modeled as various types of connecting edges.

We aim to model multiple **distinct but federating** Communities, hence distinguishing between Actors local to a community and remote Actors that live and act in different Communities. The mathematical objects and their numerical properties (the graph node and edge weights) will characterize entities within that conceptual frame. There is significant flexibility as to what decentralized network nodes and edge weights might represent. For example, weighted edges might encode information flow rate (e.g., in bits / second) between different Actors and Communities. These rates can be interpreted as *averages* but the very same values might be considered as parameters of a stochastic (probabilistic) model [19].

Pictorially, the general structure of a fully developed decentralized network might look like the figure below. Such a *heterogeneous* network enables different types of participating nodes. Heterogeneity implies that the type and amount of Actor connectivity and objects that diffuse over the network varies.



In the figure individual Actors are not shown (in practice one or more Actors will be registered within each node). This figure frames the level of our ambition around the mathematical representation of the network in terms of adjacency tensors: it is slightly more than a simplest graph representation of the network, but substantially far from a detailed model of everything that is happening in terms of information flow.

The principal technical element we need to model heterogeneity mathematically is the introduction of *network layers*. The key layer aspect that we will focus here (as a prototype for other possible uses) is

the **Community**, namely a collection of Actors that is identified by e.g., the platform they use and/or their prevalent type of activities. A fundamental assumption we will make is that Actors only live within one Community but can - in principle - interact with Actors from any other Community. This translates mathematically into studying *layer-disjoint networks*. Actors are not necessarily *Persons*. In principle we can imagine a variety of Actor categories, e.g., groups of people, algorithmic bots etc.[1] Ultimately the disjoint layers of the model express *disjoint control* over the network information flow, which may materialize in various quantitative forms.

1.3 Structure of the White Paper

In Section (2) we outline the building blocks of the methodology: the definition of layer-disjoint networks, the basic adjacency tensor, in/out-degree tensors, edge and node weights etc. We focus on the elements that are important for establishing useful network representations and mathematical graphs.

In Section (3) we dive into a very important practical aspect, namely the correspondence between the rank-4 tensors that define our heterogeneous Social Web idealizations and so-called supra-adjacency matrices. This correspondence enables using readily available results from matrix / linear algebra and the large body of existing software tools.

Finally, Section (4) further develops the tensor formalism towards calculations of dynamic properties such as how information propagates over the network. The focus here is on defining suitable graph Laplacians and the associated diffusion / random walk phenomena.

2 Building Blocks

Our previous White Paper [10] and the references therein describe background mathematical knowledge of multilayer networks in some detail. Here we will dive directly into the additional / differing elements that we want to explore in the context of heterogeneous social network structures.

The description of all multilayer networks starts with a set of graph nodes, denoted V, and a set of various possible connections between them, denoted E.[2] As sketched already in Section 1.2, in the present paper we will adopt as a canonical representation of a heterogeneous decentralized network a graph, where nodes represent Actors and edges capture their interrelationships. In addition we introduce network *aspects*, certain qualities that encode distinct attributes or behaviors of certain groups of nodes. A general multilayer network can have any number d of such differentiating aspects, enabling the modeling of very rich network structures.

The overall collection of nodes can then be indexed as follows:

$$V = \{v^{i,l_1,\ldots,l_d}\} \tag{1}$$

Thus, network nodes are identified not just by a single index i, but require an entire set of aspect indices.

[1] Distinguishing between these categories is possible but it complicates the topological properties of the network model and will not be pursued here.

[2] What is a node and what is an edge / relation is more fluid than one might think. E.g., in knowledge management so-called *reification* is a common procedure where edges (relationships between things) can be turned into nodes (so that they can subsequently be referenced and further linked to!).

2.1 Layer-disjoint Networks

Per assumption, we will partition the node set V into *disjoint subsets*, such that each node belongs to exactly one layer. These networks are called **layer-disjoint** or heterogeneous in the seminal multilayer network review paper [13]. In our case this assumption translates into any Actor belonging to only one Community. This is an important specialization of the general multilayer framework, and has implications both for its representation and for practical calculations.[3]

The heterogeneity enabled by introducing aspects implies that the type of Actors, the degree of their connectivity, the nature of their network activity (e.g., what data objects they exchange) may vary significantly depending on the Community they belong to. For simplicity in this paper we will restrict the discussion to only one such aspect, only one heterogeneity factor. Since we focus on mathematical tools we will avoid a precise specification of the aspect. It simply represents a grouping of Actor nodes that belong to a distinct *Community*. The implication is that relevant qualitative and/or quantitative properties characterizing nodes and edges in such an inhomogeneous network will vary between Communities in systematic ways that deserve capturing.

We will use Latin alphabet letters $\{i, j, k, \dots\}$ to indicate individual Actor nodes within a community and Greek alphabet letters $\{\alpha, \beta, \gamma, \dots\}$ to indicate Community instances. Thus the set of nodes is now indexed with a pair of integers as:

$$V = \{(\alpha, i)\} \tag{2}$$

Importantly, the number of Actor nodes per community will vary, potentially by a lot. We therefore must enumerate the nodes of each Community instance explicitly. The set of nodes within a Community α is $V^\alpha = \{(\alpha, i)\}_{i=1}^{n^\alpha}$, where the order $|V^\alpha| = n^\alpha$ denotes the total number of Actors in that Community. The total number of nodes in a network with m layers/communities is simply the sum:

$$n = \sum_{\alpha}^{m} n^\alpha \tag{3}$$

The distribution of node counts over layers plays an important role in the specification of the network. It will be useful for the sequel to capture that distribution also as a **partition** Π of the global index set $\{1, 2, \dots, n\}$, into m consecutive intervals of sizes $n^1, n^2, \dots, n^m$. Using these size parameters we can already express a metric that characterizes how balanced the distribution of Actors over Communities [20]. This is a vector measuring the distribution of Community sizes across the network:

$$f^\alpha = \frac{n^\alpha}{n} \tag{4}$$

2.2 Actor Connections and the Adjacency Tensor

Graph edges describe relations, interactions or associations between pairs of nodes. An edge set E of the multilayer network is a set of pairs of possible combinations of Actor nodes $(\alpha, i), (\beta, j)$. Edges can exist both *intra-layer* (e.g., within an online Community) and *inter-layer* (across Communities). When relationships between Actors are symmetric (bidirectional) they are described most economically with an *undirected graph* where the edges have no directionality or arrows. For our modeling purposes a

[3] Given that Actors are not the same as Persons, human users of decentralized networks may have *accounts* in multiple communities. Each one of those might be treated as a different Actor.

directed graph approach is much more flexible. It allows for a richer set of possible relations that in many cases is an essential requirement.[4] Bidirectional relations can be captured using two links with reverse directionality. We don't allow for more than one edge in each direction, neither do we allow for self-loops (Actor self-activation). Technically these constraints allow us to work with the more concise *adjacency matrix* concepts instead of the more flexible but also more verbose *incidence matrices*.

The linear algebra correspondence of graphs is very important for quantitative analysis. For simple graphs this reduces to a correspondence to matrices. In our multilayer context this correspondence entails defining certain higher-dimensional *tensor objects*. Tensors are direct generalizations of more familiar vectors and matrices.[5] In particular, we will use *Adjacency Tensors* that are generalizing the concept of an adjacency matrix. We write explicitly an adjacency tensor (its elements) as:[6]

$$A^{\beta j}_{\alpha i} \tag{5}$$

where the indices (i, j) range over $([1, \dots, n^{\alpha}], [1, \dots n^{\beta}])$ respectively, and the layer indices range over $[1, \dots, m]$. This is the binary adjacency tensor we introduced in WP15 [10] as a simple model of Activity-Pub networks [1]. It contains only zeros and ones. The placement of indices (upper or lower) is important, as it indicates the directionality of relations between Actors and Communities. The covariant, or lower indices (α, i) represent a *source* Actor i residing in source Community α. The contravariant, or upper indices (β, j) represent a *destination* Actor j living in Community β. This multilayer adjacency tensor (A) can represent a wealth of complicated relationships among Actors residing in different Communities, depending on what meaning we attribute to the linkages. The detailed definition of its elements is:

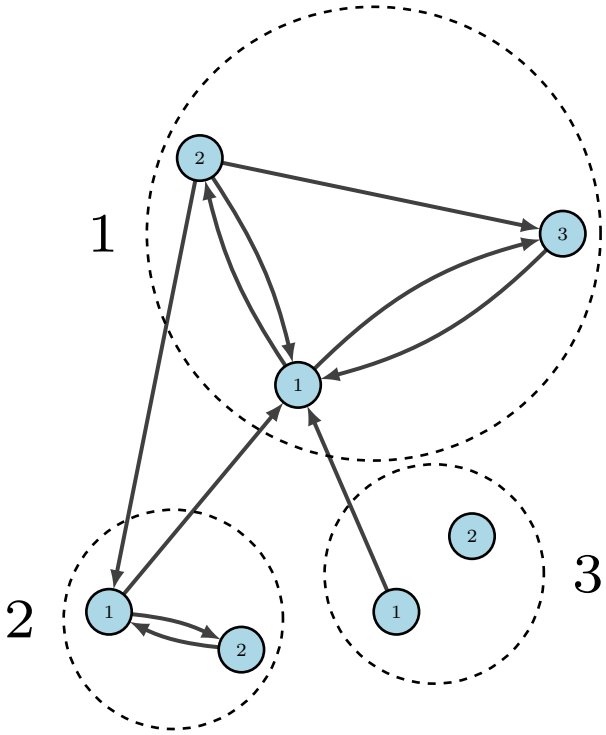
$$A^{\beta j}_{\alpha i} = \begin{cases} 1 & : \text{ there is a link from Actor } (\alpha, i) \text{ to Actor } (\beta, j) \\ 0 & : \text{ otherwise} \end{cases} \tag{6}$$

For example, tensor elements like A^{1j}_{1i}, capture relations within a certain "community-1", linking some Actor i to some Actor j. In contrast, tensor elements like A^{2j}_{1i} capture relations emanating from community-1 towards community-2. The adjacency tensor is pictorially represented as a collection of edges between Actors in different Communities:

[4] As is typically the case, increased flexibility has trade-offs: As we will see later on, directed graphs typically avail of fewer mathematical tools hence when the research question does not require it, it might be opportune to work with the underlying undirected graph.

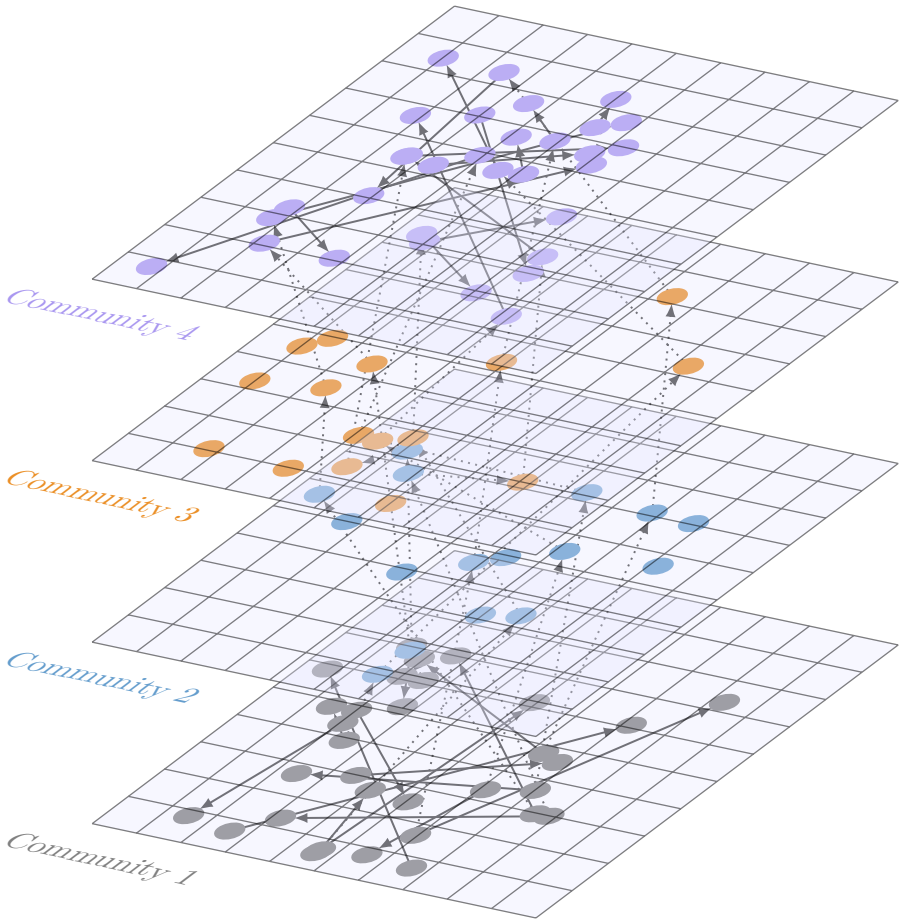
[5] Vectors could be called rank-1 tensors, indicating that they have one index. Matrices are correspondingly rank-2 tensors, they have two indexes or dimensions. The higher number of indices, the higher the rank of a tensor.

[6] We avoid the more heavy coordinate free notation used in the multilayer literature as it does not add anything to the present discussion.



The above diagram sketches a decentralized Social Web network comprising three Communities with varying numbers of Actors in each. The linkages are directional (from source Actors to target Actors). They may model Follower/Followee relationships or any other meaningful peer relation in the online network context. Actors reside exclusively within one Community but can link (interact) with Actors in any other Community. Notice that the index over Actors in different Communities has different ranges (Thee partition Π here has intervals $n^1 = 3, n^2 = 2, n^3 = 2$) and Actors with the same numerical index value but living in different Communities need not have any relation with each other!

As the number of Actors and Communities increases (as per the figure below), it becomes harder to visualize the network structure. E.g., if treat different Mastodon server instances as layers/Communities, the layers run currently in the multiple thousands. Mastodon accounts (Actors) are interacting both within their Community (in Community 1 all such intra-layer connections are shown as solid lines) and across Communities (dotted lines linking between layers). Another modeling approach is to treat as a Community each collection of servers running distinct software platforms (e.g., the collection of servers running PeerTube, FunkWhale, Mobilizon etc. implementations of the ActivityPub software).



2.3 Actor In/Out-neighbors

Out-neighbors is the set of Actors that a given Actor points towards (has outgoing links) to. For example, the set could be established from the "follower" relationships of other Actors, implying that information flows *from* that Actor *to* these other Actors. The set of all destinations from the source Actor (α, i) is denoted as N^+ or N_{out}:

$$N^+(\alpha, i) = \{(\beta, j) \mid A_{\alpha i}^{\beta j} = 1, \beta \in [1 \dots m], j \in [1, \dots n^\beta]\} \tag{7}$$

The *In-neighbors*, is the set of source Actors that point toward a given Actor. They would be for example the set of all Actors being "followed" by a given Actor. They are denoted correspondingly as N^- or N_{in}:

$$N^-(\alpha, i) = \{(\beta, j) \mid A_{\beta j}^{\alpha i} = 1, \beta \in [1 \dots m], j \in [1, \dots n^\beta]\} \tag{8}$$

If we group all immediate neighbors regardless of direction, we have the union $N = N^+ \cup N^-$. These sets are the basis for calculating some core objects in graph science, as we will see next. In a multilayer network it is also meaningful to isolate in/out neighbors within the same layer and across layers. For example, if we want to isolate outgoing links from actor (α, i) to a specific layer γ we need:

$$N_\gamma^+(\alpha, i) = \{(j, \beta) \mid A_{\alpha i}^{\beta j} = 1, \beta = \gamma, j \in [1, \dots n^\gamma]\} \tag{9}$$

2.4 In/Out Degree Tensors

The number of in/out connections between Actors in networks is one of the most studied quantitative properties of social networks, giving rise to new concepts such as small-world networks [21]. These metrics determine e.g., the degree to which the network is diversified and balanced (a peer-based network) or highly concentrated (an *influencer* dominated network). Virality and other such collective phenomena are implicated, though several other network aspects might be involved [22].

The *global in-degree* of an Actor (α, i) is simply the count of all j edges ending with them, irrespective of the originating Community layer β. The result of performing this counting across the network is a rank-2 contravariant tensor:

$$\check{D}^{\alpha i} = \sum_{\beta}^{m} \sum_{j}^{n^{\beta}} A^{\alpha i}_{\beta j} \tag{10}$$

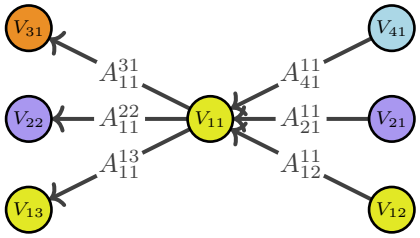
where the check mark $\check{D}$ is a mnemonic to indicate that we are summing over edges going *into the node*. Recall here that the upper indices are the target node/layers, the lower indices source node/layers. Notice that *the order of the summation operations matters*: it must be done *layer-wise* first, so that the inner sum "knows" how many source nodes (β, j) exist within each layer β (the value of n^{β} changes for each β).

The corresponding *global out-degree* edge count of the same Actor is the count of all j edges emanating in (α, i), irrespective of which Actor j in Community β they connect to.

$$\hat{D}_{\alpha i} = \sum_{\beta}^{m} \sum_{j}^{n^{\beta}} A^{\beta j}_{\alpha i} \tag{11}$$

where the hat indicates summing over edges *emanating* from the node.

As an example, consider a four-layer network where the layers $1, 2, 3, 4$ are represented also via color codes as below:



The non-zero elements of the adjacency tensor along with their indices are overlaid on the edges. Our focus node $(1, 1)$ is at the center, and is part of layer 1 (light green). It has three incoming edges, thus its in-degree is $\check{D}^{11} = 3$ and three outgoing edges, thus its out-degree is $\hat{D}_{11} = 3$.

2.4.1 Rank-4 In/Out-Degree Tensors

The rank-2 degree tensors we have computed above can be *padded* into rank-4 tensors by using special tensors, the *Kronecker deltas*. This procedure is the equivalent of constructing a diagonal matrix from of a vector, by placing all vector values along the matrix diagonal and setting everything else to zero. Here this can be accomplished using the identity tensor $I = \delta^{\beta j}_{\alpha i}$, defined as:

$$\delta_{\alpha i}^{\beta j} = \begin{cases} 1 & : \text{ if } (i = j) \text{ and } (\alpha = \beta) \\ 0 & : \text{ otherwise} \end{cases} \tag{12}$$

With a simple *element-wise* multiplication of this tensor with the in/out degree tensors we obtain:

$$\hat{D}_{\alpha i}^{\beta j} = \hat{D}_{\beta j} \delta_{\alpha i}^{\beta j} \tag{13}$$

and

$$\check{D}_{\beta j}^{\alpha i} = \check{D}^{\beta j} \delta_{\beta j}^{\alpha i} \tag{14}$$

These tensors don't encode any new information but their form will make them useful in the sequel.

2.4.2 Local versus Remote Degrees

How do the global connectivity degrees of an Actor (those involving all layers) decompose into those within their own layer and the rest? It is an exercise in splitting the sum into intra and inter-layer sums, which we can do by introducing another Kronecker delta, δ_{β}^{α}:

$$\delta_{\alpha}^{\beta} = \begin{cases} 1 & : \text{ if } (\alpha = \beta) \\ 0 & : \text{ otherwise} \end{cases} \tag{15}$$

For example, the global out-degree of a node splits into:

$$\hat{D}_{\alpha i} = \sum_{\beta}^{m} \sum_{j}^{n^{\beta}} A_{\alpha i}^{\beta j} \tag{16}$$

$$= \sum_{\beta}^{m} \sum_{j}^{n^{\beta}} A_{\alpha i}^{\beta j} (1 - \delta_{\beta}^{\alpha} + \delta_{\beta}^{\alpha}) \tag{17}$$

$$= \sum_{\beta}^{m} \sum_{j}^{n^{\beta}} A_{\alpha i}^{\beta j} (1 - \delta_{\beta}^{\alpha}) + \sum_{\beta}^{m} \sum_{j}^{n^{\beta}} A_{\alpha i}^{\beta j} \delta_{\beta}^{\alpha} \tag{18}$$

$$= \sum_{\beta}^{m} \sum_{j}^{n^{\beta}} A_{\alpha i}^{\beta j} (1 - \delta_{\beta}^{\alpha}) + \sum_{j}^{n^{\alpha}} A_{\alpha i}^{\alpha j} \tag{19}$$

The first term captures connectivity with Actors in other Communities and the second term ($\sum_{j}^{n^{\alpha}} A_{\alpha i}^{j\alpha}$) connectivity within the same Community.

2.4.3 Counting all Connections

The total number of connections in the network is the formidable sum over all Communities and all Actors:

$$e = \sum_{\alpha}^{m} \sum_{\beta}^{m} \sum_{i}^{n^{\alpha}} \sum_{j}^{n^{\beta}} A_{\alpha i}^{\beta j} \tag{20}$$

While conceptually straightforward and clear, the summation notation can become quite unwieldy when working with high-order tensor expressions. This is where sometimes the *Einstein summation convention* helps restore some brevity. The Einstein summation rule means that repeating pairs of indices *implies* a

sum over those indices, which permits dropping the summation signs. Alas, for the Einstein rule to be usable as a general convention in all expressions, indices must *have the same range*. In relativistic theories of physics, where this notation was established, this range is over time and space dimensions $0, 1, 2, 3$ that apply similarly to all physical quantities. For our layer-disjoint network the layer indexes are indeed universal (all layer summations are over the $1, \ldots, m$ range), but the node indices have different ranges ($n^1, \ldots, n^m$ per layer). This reflects that, in general, there is a different Actor count per Community. Unfortunately this means that a contraction expression that involves node aggregation over different layers is ambiguous. Consider, for example, using the Einstein rule to represent the contraction of two tensors A and U, $A^{\beta j}_{\alpha i} U^{\delta i}_{\gamma k}$ along a pair of covariant / contravariant node indices (i). We have α and δ as fixed layers and would like to sum over node connections between them. But the Einstein summation convention does not (and cannot) clarify whether the summation is over the n^a or n^δ range of the i index.[7]

$$\sum_{i}^{n^\alpha} A^{\beta j}_{\alpha i} U^{\delta i}_{\gamma k} \neq A^{\beta j}_{\alpha i} U^{\delta i}_{\gamma j} \neq \sum_{i}^{n^\delta} A^{\beta k}_{\alpha i} U^{\delta i}_{\gamma j} \tag{21}$$

While we cannot achieve the brevity of the Einstein convention (and we will avoid the introduction of awkward "shadow" nodes just for the purpose of making the ranges uniform), the exercise gives us an opportunity to at least somewhat simplify the notation, by consolidating sums over layer/nodes as follows:

$$e = \sum_{\alpha i}^{mn^\alpha} \sum_{\beta j}^{mn^\beta} A^{\beta j}_{\alpha i} (1 - \delta^\alpha_\beta + \delta^\alpha_\beta) \tag{22}$$

$$= \sum_{\alpha i}^{mn^\alpha} \sum_{\beta j}^{mn^\beta} A^{\beta j}_{\alpha i} (1 - \delta^\alpha_\beta) + \sum_{\alpha i}^{mn^\alpha} \sum_{\beta j}^{mn^\beta} A^{\beta j}_{\alpha i} \delta^\alpha_\beta \tag{23}$$

$$= \sum_{\alpha i}^{mn^\alpha} \sum_{\beta j}^{mn^\beta} A^{\beta j}_{\alpha i} (1 - \delta^\alpha_\beta) + \sum_{\alpha i}^{mn^\alpha} \sum_{j}^{n^\alpha} A^{j\alpha}_{\alpha i} \tag{24}$$

The grand total can thus be decomposed into intra-layer and inter-layer edge counts as one might expect.

2.5 Weighted Adjacency Tensors

So far we worked with the digital (0, 1) adjacency tensor A. A generalization we make to avail of additional modeling flexibility is to introduce *real-valued tensors*, typically positive real values (strengths or weights). Thus, instead of tensor values restricted to 1 or 0 (indicating that there is or there isn't a connection between Actors) the tensor elements may have arbitrary values. Such values might indicate the bandwidth, capacity, strength of affiliation, or any other numerically quantifiable property of a connection that characterizes the relationship and can be expressed as a scalar.[8]

A fourth-order tensor $W^{i\beta}_{i\alpha}$ that encodes positive real numbers (called a *weighted adjacency tensor*), can encode directed, weighted links from Actor i in Community α to node j in Community β. This is expressed as:

[7] In passing, we also note that given a layer-disjoint network with different number of nodes per layer, a Kronecker delta δ^i_j with indices ranging of nodes in layers is also not well defined.

[8] While in principle edge values can be negative, in a directed graph context negative values can be more opportunely expressed by a reversal of the edge direction.

$$W_{\alpha i}^{\beta j} = \begin{cases} w_{\alpha i}^{\beta j} > 0 & : \text{ weight of link from node } (i,a) \text{ to node } (j,b) \\ 0 & : \text{ otherwise} \end{cases} \tag{25}$$

We are free to introduce *multiple* such weighted adjacency tensors, encoding any distinct connection attributes of relevance for the analysis. For example, a weighted adjacency tensor could be counting *reactions* such as "likes" or "reposts" by Actor (i,α) to content distributed by Actor (j,β). Strictly speaking any such new tensor is a new graph, but, given it utilizes the same number of layers and the same partition, it is *conformant*, which, as we will see, makes it easily usable alongside any other such tensors.

Weighted graphs are generalizations of the unweighted case. In a sense the binary adjacency tensor A is still encoded in the values of W: It requires applying a *sign function* to extract:

$$A_{\alpha i}^{\beta j} = \begin{cases} 1 & : \text{ if } W_{\alpha i}^{\beta j} > 0 \\ 0 & : \text{ otherwise} \end{cases} \tag{26}$$

2.5.1 In and Out-strength of Nodes

In the weighted adjacency scenario, the edge counts leading to in/out degrees are replaced by the concepts of *out-strength* and *in-strength* of a node, which sum the weights of the connections. The formulas are identical to the unweighted case, with W simply replacing A.

$$\check{S}^{\alpha i} = \sum_{\beta j}^{mn^{\beta}} W_{\beta j}^{\alpha i} \tag{27}$$

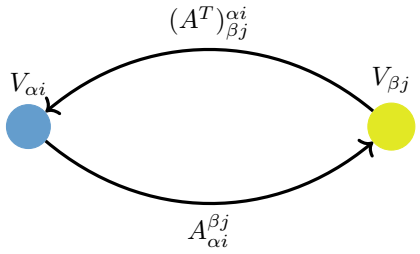
$$\hat{S}_{\alpha i} = \sum_{\beta j}^{mn^{\beta}} W_{\alpha i}^{\beta j} \tag{28}$$

2.6 The Transpose or Reverse Graph

A graph that is sometimes useful to consider is the transpose or reverse graph, where *every* directed edge of the original graph is flipped, pointing exactly along the opposite direction. The transpose graph has the same number of nodes and layers as the original graph (conformant) and also the same total number of edges, both within and across layers. But the notion of in/out degrees get swapped.

Using index notation,

$$(A^T)_{\beta j}^{\alpha i} = \begin{cases} 1 & : \text{ if } A_{\alpha i}^{\beta j} = 1 \\ 0 & : \text{ otherwise} \end{cases} \tag{29}$$



2.7 Node Weighted Graphs

We have established that fourth-order, rank-4, adjacency tensors like A or W enable representations of heterogeneous Social Web networks that involve distinct groupings in Communities. Further modeling flexibility is afforded with the introduction of flexible *node weights*. Introducing node weights entails *labeling* network Actors with numerical values. Actor specific node weights could be e.g., the rate or posting by a publishing Actor, their frequency of accessing content, the frequency of forwarding content to one's connections, or any other attribute that characterizes individual Actors *without any reference to other Actors*.

For classic (simple) graphs, node properties would be vectors over the node space, e.g., represented as vector B_j. For multi-layer networks, node properties become matrices (or even higher-order tensors, depending on the number of aspects being modeled), as they may depend on the layers on which nodes reside. Introducing node weights brings our network model into the realm of *double-weighted graphs*. Traditional graph theory focuses primarily on edge-weighted graphs, but doubly weighted graphs are also a formal, recognized part of graph theory. Other names for these objects are *vertex-and-edge-weighted graphs* or *labeled graphs*.

2.7.1 Covariant and Contravariant Weights

In our canonical network model that models Actors within their Communities, a rank-2 object such as $B^{\alpha i}$ encodes a property of Actor i in Community α. Alternatively we could also use an object with lower indices, namely $B_{\alpha i}$. The difference between these choices becomes apparent in specific use contexts, when we combine tensors to express e.g., the propagation of information over the network. In certain applications (Random Walks, Advection-Diffusion, or PageRank type calculations), a node state is represented as a **contravariant** vector or tensor. The node weight might represent intensity, probability, or some node state value. In supra-adjacency context it acts as a *column vector*, representing an evolving state. Roughly speaking, the directed edges act to "move" or update information from a source node to a target node. An adjacency tensor in this picture acts as a *linear operator* that takes the network state at a source node, transforms it, and maps it to a target node. To update a state by *pushing forward* along the directed network edges, we contract the upper indices of the vector with the lower (source) indices of an adjacency tensor:

$$B^{\beta j}_{t+1} = \sum_{\alpha i}^{mn^{\alpha}} A^{\beta j}_{\alpha i} B^{\alpha i}_{t} \tag{30}$$

Alternatively, a vector can be represented as a **covariant** vector or tensor. In that case it features two lower indices. This represents a *linear functional* and it acts as a *row vector*. To see how weights propagate backward, or to calculate a weighted output (e.g. forming a consensus from inputs), we contract the lower indices of the vector with the upper (destination) indices of an adjacency tensor.

$$B^{s+1}_{\alpha i} = \sum_{\beta j}^{mn^{\alpha}} B^{s}_{\beta j} A^{\beta j}_{\alpha i} \tag{31}$$

3 Matrix Representations

3.1 The Supra-Adjacency Matrix

The tensor notation we used thus far is succinctly expressing complex graph relations in heterogeneous networks. Various numerical libraries can directly manipulate tensorial objects [23], but for a wider choice it might be advantageous to flatten or roll-out the tensors into matrices and vectors. This exercise is particularly fruitful for layer-disjoint networks. While general tensors do not strictly have "rows" and "columns", unless they are rank-2 (matrices), in layer-disjoint case we can think of the lower indices as a generalized row index and the upper indices as a generalized column index. This relation of co/contravariant indices to rows and columns will become very explicit when we study the equivalence of tensor objects with matrices. In that operation the contra/covariant indices will be combined to form two matrix-like supra-indices.

The *supra-adjacency matrix* is a block matrix of size $(n \times n) = (\sum_\alpha^m n_\alpha \times \sum_\alpha^m n^\alpha)$, where rows indicate the source layer and columns indicate the target layer of a graph relation. The supra-adjacency matrix in block form is:

$$\mathbf{A}_\alpha^\beta = \begin{pmatrix} \mathbf{A}_1^1 & \mathbf{A}_1^2 & \cdots & \mathbf{A}_1^m \\ \mathbf{A}_2^1 & \mathbf{A}_2^2 & \cdots & \mathbf{A}_2^m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_m^1 & \mathbf{A}_m^2 & \cdots & \mathbf{A}_m^m \end{pmatrix} \tag{32}$$

In the above, diagonal blocks encode intra-layer connectivity and off-diagonal blocks encode inter-layer connectivity. The row index α denotes the source layer. Every block in row α contains edges originating exclusively from layer α. The column index β denotes the target layer. Every block in column β contains edges terminating exclusively in layer β. The diagonal blocks $\mathbf{A}_\alpha^\alpha$ are square matrices with dimensions $n^\alpha \times n^\alpha$. The off-diagonal blocks are rectangular matrices $\mathbf{A}_\alpha^\beta$ with dimensions $n^\alpha \times n^\beta$. This means that off-diagonal blocks in transposed positions have their row / column sizes swapped (e.g., the size of block $\mathbf{A}_\beta^\alpha$ is $n^\beta \times n^\alpha$).

The unfolding of the adjacency tensor for our small graph example would look as follows:

$$\mathbf{A} = \left[\begin{array}{ccc|ccc|cc} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{33}$$

3.2 Symmetrically Partitioned Matrices

The supra-adjacency matrices representing layer-disjoint networks are a specific type of *square block matrices*. The broader set of (general) square $(n \times n)$ real valued matrices $\mathbf{M}_n^n(R)$ form an associative algebra over R. Our focus here is the subset of matrices $\mathbf{M}_n^n(R, \Pi)$ of symmetrically partitioned block matrices of size $n \times n$ and having a partition Π. Any matrix $\mathbf{A} \in M_n^n(R, \Pi)$ is split into blocks $\mathbf{A}_\alpha^\beta$ (as per Eq. 32).

Such symmetrically partitioned matrices $\mathbf{M}_n^n(R, \Pi)$ with a fixed partition schedule Π form a subalgebra of the square matrix algebra. A notable aspect is that they remain a closed space under standard matrix

multiplication. In other words a product matrix $\mathbf{AB}$ of two symmetrically partitioned block matrices is also a symmetrically partitioned matrix, of the same dimensions $n \times n$, and with the same partition Π.

When calculating the product $\mathbf{C} = \mathbf{AB}$, the resulting blocks $\mathbf{C}_\alpha^\beta$ have the dimensions $n^\alpha \times n^\beta$ as the corresponding $\mathbf{A}_\alpha^\beta$ and $\mathbf{B}_\alpha^\beta$ blocks. The product sum can be expressed both at matrix element level and at matrix block level:

$$C_{IJ} = \sum_{K=1}^{n} A_{IK} B_{KJ} \tag{34}$$

$$\mathbf{C}_\alpha^\beta = \sum_{\gamma=1}^{m} \mathbf{A}_\gamma^\beta \mathbf{B}_\alpha^\gamma \tag{35}$$

$$\tag{36}$$

3.3 From Tensors to Matrices and back

The supra-adjacency matrix and tensor representations of layer-disjoint multilayer networks are essentially equivalent. For any symmetrically partitioned matrix $\mathbf{A}$, we can uniquely construct a layer-disjoint network adjacency tensor $\mathcal{A}$ by defining the intra-layer edges of layer α using the diagonal blocks $\mathbf{A}_\alpha^\alpha$, and the cross-layer connections between layer α and layer β using the off-diagonal blocks $\mathbf{A}_\alpha^\beta$.

To show this more explicitly we need a one-to-one mapping that translates the tensor indices (i, j, α, β) to unique supra-adjacency matrix row and column indices (I, J). Vice-versa, we want for each row index I and column index J of the supra-adjacency matrix $\mathbf{A}$ to reconstruct the node and layer indices (i, j, α, β) of the adjacency tensor A. To facilitate the presentation we introduce the *partial Offset sums* for the given partition Π:

$$\text{Offset}(\alpha) = \sum_{\gamma=1}^{\alpha-1} n^\gamma \quad (\text{with Offset}(1) = 0) \tag{37}$$

The mapping for individual nodes within their layers to matrix positions is:

$$I(\alpha, i) = \sum_{\gamma}^{\alpha-1} n^\gamma + i \quad \text{where } 1 \leq i \leq n^\alpha \tag{38}$$

$$J(\beta, j) = \sum_{\gamma}^{\beta-1} n^\gamma + j \quad \text{where } 1 \leq j \leq n^\beta \tag{39}$$

$$\tag{40}$$

where under the assumption $n^0 = 0$ we also cover the first blocks (α and/or β being equal to 1).

In matrix index notation we write the elements of the supra-adjacency matrix as A_{IJ}. Using these index transformations we assign elements to the supra-adjacency matrix from the corresponding tensor elements. The link between matrix indices (I, J) and tensor indices $(\alpha, i), (\beta, j)$ is given by:

$$A_{IJ} = A_{I(\alpha,i)J(\beta,j)} = A_{\alpha i}^{\beta j} \tag{41}$$

For the inverse calculation, to find which layer a matrix row index I belongs to, we look for the unique layer index α such that I falls within that layer's block boundaries. Once the layers α and β are identified, node indices within layers are the remainder after subtracting the Offsets:

- $\alpha(I,J)$ is the unique integer in $\{1,\ldots,m\}$ satisfying $\text{Offset}(\alpha) < I \leq \text{Offset}(\alpha+1)$
- $\beta(I,J)$ is the unique integer in $\{1,\ldots,m\}$ satisfying $\text{Offset}(\beta) < J \leq \text{Offset}(\beta+1)$
- $i(I,J) = I - \text{Offset}(\alpha)$
- $j(I,J) = J - \text{Offset}(\beta)$

With those four functions, for any given (I,J) pair we obtain the $(\alpha,i),(\beta,j)$ tuple, which in turn helps define the elements of the adjacency tensor:

$$A^{\beta j}_{\alpha i} = A^{\beta(I,J)j(I,J)}_{\alpha(I,J)i(I,J)} = A_{IJ} \tag{42}$$

3.4 Graph Operations

3.4.1 Transpose of Supra-Adjacency Matrix

The transpose operator is a simple example of a large collection of possible *graph operations*. In supra-adjacency representation, the transpose graph we introduced before has a transposed supra-adjacency matrix:

$$(\mathbf{A}^T)^{\beta}_{\alpha} = \begin{pmatrix} (\mathbf{A}^1_1)^T & (\mathbf{A}^1_2)^T & \cdots & (\mathbf{A}^1_m)^T \\ (\mathbf{A}^2_1)^T & (\mathbf{A}^2_2)^T & \cdots & (\mathbf{A}^2_m)^T \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{A}^m_1)^T & (\mathbf{A}^m_2)^T & \cdots & (\mathbf{A}^m_m)^T \end{pmatrix} \tag{43}$$

with transposed block positions. This means that e.g., $(A^2_1)^T$ takes the position of A^1_2, but we also transpose the blocks internally.

3.4.2 Symmetrization and Anti-symmetrization

Every square matrix can be uniquely decomposed into a symmetric part and an antisymmetric (skew-symmetric) part. Unlike matrices, there isn't a single universal decomposition into symmetric and antisymmetric part for general rank-4 tensors. But the supra-adjacency mapping suggests we can always decompose layer-disjoint graphs into symmetric and antisymmetric parts. This is possible because the node and layer indices are always operated on together (it does not make sense, e.g., for layer-disjoint adjacency tensors to swap node indices with layer indices).

Thus, any layer-disjoint adjacency tensor be decomposed as follows:

$$\mathbf{A} = \frac{1}{2}(\mathbf{A} + \mathbf{A}^T) + \frac{1}{2}(\mathbf{A} - \mathbf{A}^T) \tag{44}$$

$$A^{\beta j}_{\alpha i} = \frac{1}{2}(A^{\beta j}_{\alpha i} + (A^T)^{\beta j}_{\alpha i}) + \frac{1}{2}(A^{\beta j}_{\alpha i} - (A^T)^{\beta j}_{\alpha i}) \tag{45}$$

In the next section we will briefly touch on the ramifications of such decompositions.

3.4.3 Number of Walks between Nodes

The k-th power of the supra-adjacency matrix counts the number of walks of length k between any two Actors in the network (irrespective of Community).

$$\mathbf{A}^k = \mathbf{A} \dots \mathbf{A}, \text{k times} \tag{46}$$

Just as the supra-adjacency matrix itself gives the number of direct edges (walks of length 1) between Actors (α, i) and (β, j), the second power gives the number of walks of length 2, that is, paths using one intermediate node between two Actors. The k-th power $\mathbf{A}^k$ provides the number of distinct k-step paths between two Actors.

NB: While all powers of $\mathbf{A}$ correspond to new graphs, these are no longer binary graphs with unweighted edges.

3.4.4 Breath-first Search and Semiring Extensions

A common operation involving graphs is *breadth first search* (BFS). In our context a breadth-first-search algorithm would return the set of Actors reachable from Actor (α, i), following the direction of their connections (follows). The corresponding tensor operation is to contract the adjacency tensor A of the graph by a vector u of size n that has a single entry (unity) corresponding to the starting Actor (α, i).

As before, using powers of the supra-adjacency matrix, we progressively probe ever wider linkages of the starting Actors. Breadth-First Search (BFS) on a heterogeneous network can thus be expressed algebraically as a sequence of matrix-vector multiplications using the supra-adjacency matrix. The subtlety is that as we expand the "influence cone" of our starting Actor, we must track Actors already reached by step k (to avoid double counting). This is accomplished using element-wise *logical operations*. In the supra-matrix formulation thus far we made use only of the conventional algebraic operations (addition and multiplication). Evaluating vector and matrix multiplications is possible in a more general algebraic setting, symbolically:

$$(A \otimes B)_{IJ} = \bigoplus_K (A_{IK} \otimes B_{KJ})$$

where *oplus* is semiring's *additive operation* and $\otimes$ is the corresponding *multiplicative operation*. Adopting various such semiring structures changes the meaning of graph traversal operations and enables expressing a wealth of more complex metrics [24].

3.5 Overture to Applications

In this section we introduced the fundamental objects or building blocks that we want to work with in our models of heterogeneous social networks, along with a notation adapted to specific layer-disjoint structure. In particular the close tensor-to-matrix equivalence established above enables the use of a variety of standard numerical software for working with network data. This is a considerable convenience as it provides both containers for storing data and a variety of useful algorithms.

In concrete applications these building blocks might be populated with data either from *empirical observations* (statistical data reported from actual operating decentralized networks) or be *postulated*, as theoretical network topologies. The first class of applications aims to understand real-life network structure and behavior, it is an ex-post analysis of what has happened. The second class of applications

aims to anticipate and predict network phenomena, scaling performance etc. It is a form of ex-ante or what-if analysis.

The simplest example of a theoretical network structure is probably the *fully connected* network, i.e., setting all edges to 1 ($A_{\alpha i}^{\beta j} = 1$) for all possible 4-tuples (i, j, α, β). This model can serve e.g., as a benchmark against which to compare the actual observed connectivity of the network. Another class of theoretical models used heavily in applications are *random models*, where the number of nodes and edges is sampled from probability distributions.

Many applications focus on distilling measures of network *concentration*. Measuring the concentration of links, the preponderance of certain connectivity patterns (motifs) etc. requires defining suitable local and global aggregates taking the adjacency tensor and any node weights as inputs. In a previous White Paper in the *Connecting the Dots* series [25] we examined many such metrics, exploring the index construction mechanics, the calculation steps required to transform graph data sets into concentration measurements and enumerated the large number of commonly used indexes, categorized according to their broad nature.

More elaborate applications, a glimpse of which we have in the next section, tend to introduce *additional assumptions* about the dynamics governing the network behavior. While these model assumptions tend to be stylized versus the actual behaviors of large numbers of interacting individuals, they offer additional tools to probe into otherwise inaccessible domains.

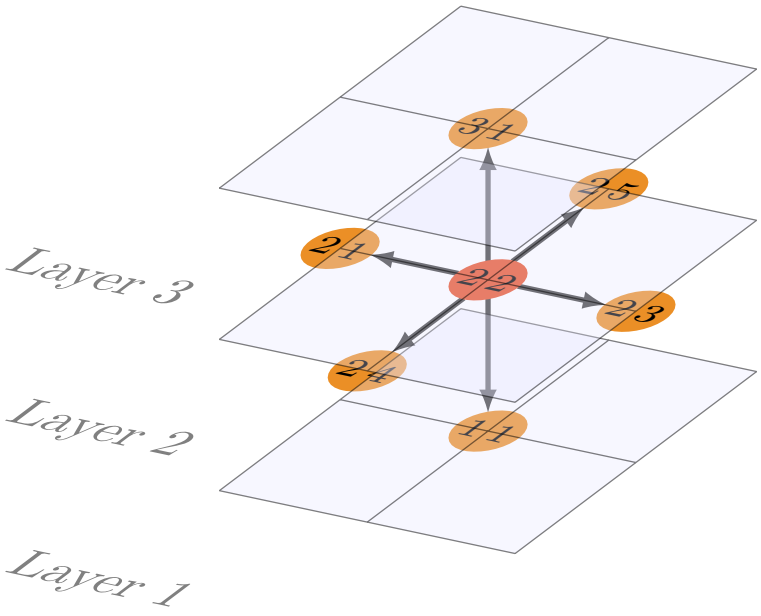
4 Graph Laplacians and Diffusion Models

In this final section we look into one of the most important objects that can derived from the adjacency tensors already introduced, namely the so-called **graph Laplacians**. The name of these objects derives from a classic operator in calculus, the *Laplacian operator*, which itself is named after the important French mathematician *Pierre-Simon Laplace*. The Laplacian is a second order differential operator, expressing the divergence of the gradient. In Cartesian coordinates and for an l-dimensional space, this operator can be written as the sum of second order partial derivatives:

$$\Delta = \nabla^2 = \sum_{k}^{l} (\frac{\partial}{\partial x^k})^2 \tag{47}$$

The first step of establishing the graph-like incarnation of the Laplacian, is to move from a continuous space to a discrete space formulation (sometimes denoted a "grid"). On a discrete space, continuous operators such as (∇ and Δ) can be *approximated* via discretized versions, with the approximation becoming increasingly more accurate as the spacing of the grid (distance between nodes) is reduced.

One way to see the transition to a discrete graph context is to examine the *finite difference discretization* of the Laplacian operator, on a regular rectangular grid. For representational convenience we take the space to be three-dimensional. We imagine that the nodes in this grid / graph are labeled by a weight function, a scalar ϕ. This might be represented by a covariant rank-2 tensor $\phi_{\alpha i}$ where $i \in [1]$ for layer 1, $i \in [1 \dots 6]$ for layer 2 and $i \in [1]$ for layer 3.



In the above three-dimensional setup and using a standard central finite difference stencil the Laplacian operator applied to a scalar function ϕ at node $2, 2$, would be proportional to:

$$\Delta\phi_{22} \sim \phi_{25} - 2\phi_{22} + \phi_{24} \tag{48}$$
$$+ \phi_{23} - 2\phi_{22} + \phi_{21} \tag{49}$$
$$+ \phi_{31} - 2\phi_{22} + \phi_{11} \tag{50}$$
$$\tag{51}$$

Notating explicitly the non-zero elements of the adjacency tensor, this expression can be rewritten as follows:

$$\Delta\phi_{22} \sim A^{11}_{22}(\phi_{11} - \phi_{22}) \tag{52}$$
$$+ A^{25}_{22}(\phi_{25} - \phi_{22}) + A^{24}_{22}(\phi_{24} - \phi_{22}) + A^{23}_{22}(\phi_{23} - \phi_{22}) + A^{21}_{22}(\phi_{21} - \phi_{22}) \tag{53}$$
$$+ A^{31}_{22}(\phi_{31} - \phi_{22}) \tag{54}$$
$$= \sum_{\beta j}^{mn^{\beta}} A^{\beta j}_{22}(\phi_{\beta j} - \phi_{22}) \tag{55}$$

where the final step sums over all target nodes in the network (including where there is no connection - the adjacency tensor has zero value).

An alternative expression follows from rearranging in terms of the out-degree $\hat{D}_{22}$ of the central node $(2, 2)$.

$$\Delta\phi_{22} \sim \sum_{\beta j}^{mn^{\beta}} A_{22}^{\beta j}(\phi_{\beta j} - \phi_{22}) \tag{56}$$

$$= \sum_{\beta j}^{mn^{\beta}} A_{22}^{\beta j}\phi_{\beta j} - \sum_{\beta j}^{mn^{\beta}} A_{22}^{\beta j}\phi_{22} \tag{57}$$

$$= -\hat{D}_{22}\phi_{22} + \sum_{\beta j}^{mn^{\beta}} A_{22}^{\beta j}\phi_{\beta j} \tag{58}$$

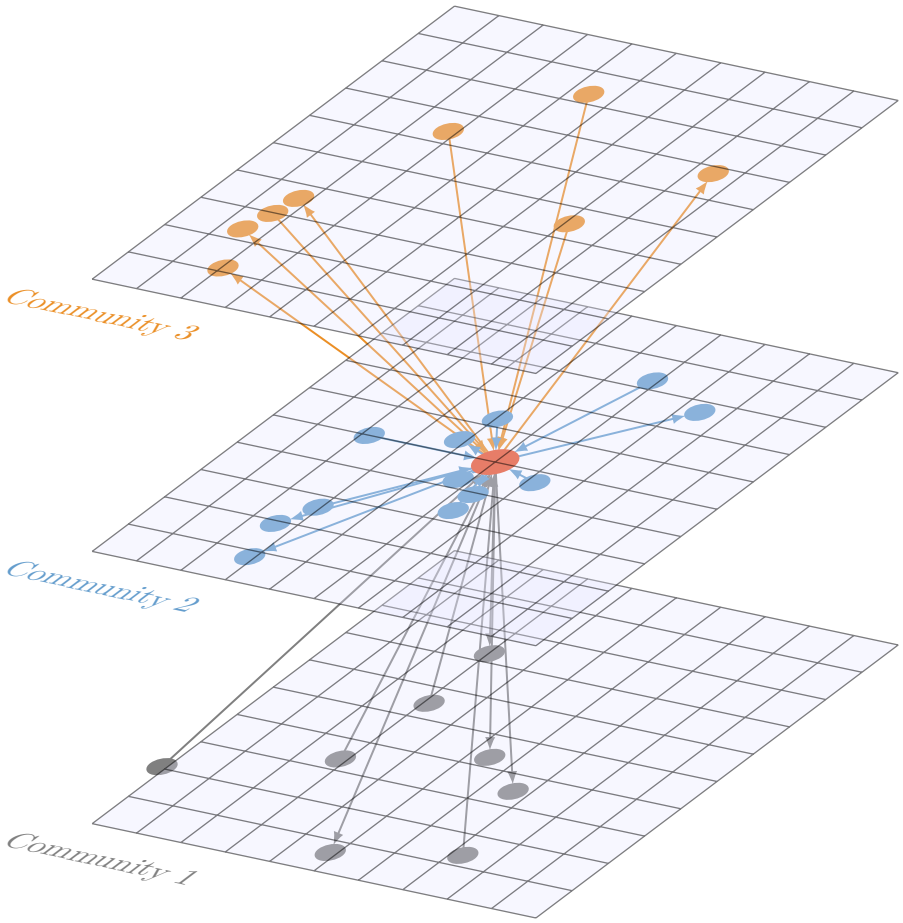
$$= -\hat{D}_{22}\sum_{\beta j}^{mn^{\beta}} \delta_{22}^{\beta j}\phi_{\beta j} + \sum_{\beta j}^{mn^{\beta}} A_{22}^{\beta j}\phi_{\beta j} \tag{59}$$

$$= -\sum_{\beta j}^{mn^{\beta}} (\hat{D}_{22}\delta_{22}^{\beta j} - A_{22}^{\beta j})\phi_{\beta j} \tag{60}$$

The term $\hat{D}_{22}\delta_{22}^{\beta j} - A_{22}^{\beta j}$ is the discrete graph operator that approximates the original continuous Laplacian operator. It is constructed from the out-degree tensor and the adjacency tensor. This concrete construction was based on the "finite difference" stencil of the standard Laplacian operator, but it paves the way to a generalization where it is applied to nearest neighbors as those are defined by the graph structure.

4.1 The Combinatorial Multilayer Graph Laplacian

In the general case the multilayer graph Laplacian centered on a node is formed from the nearest neighbors of the node, which are obtained from the adjacency tensor, alongside the degree tensor.



In contrast to the local grid Laplacian example with its implicit geometric notions of distance, scale, volume etc., in a general network the adjacency tensor may connect any given node to potentially any other node / layer. In the diagram above we display a network of three Communities, focusing on an Actor residing in the middle one. They have multiple connections both within and outside their own layer. The *combinatorial Laplacian* thus constructed involves only the adjacency tensor and its binary values, which is where the name derives from. The general expression is:

$$\hat{L}^{\beta j}_{\alpha i} = \hat{D}_{\alpha i}\delta^{\beta j}_{\alpha i} - A^{\beta j}_{\alpha i} \tag{61}$$

Building on the tensor syntax we can also write:

$$\hat{\mathcal{D}}^{j\beta}_{i\alpha} = \hat{D}_{\alpha i}\delta^{\beta j}_{\alpha i} \tag{62}$$

which produces the prototypical form of a graph Laplacian tensor:

$$\hat{L}^{j\beta}_{i\alpha} = \hat{\mathcal{D}}^{j\beta}_{i\alpha} - A^{j\beta}_{i\alpha} \tag{63}$$

Alas, edges will in general not all be outgoing and directionality generally complicates the construction of graph Laplacians. Even in our stylized discretization example, if we were to visit one of the out-neighbors of the central node they would have incoming edges, not allowing a simple replication of the procedure. While this complication can be avoided for undirected graphs, for a directed graph we are led to constructing *two types of graph Laplace operators*, depending on whether they use the outgoing or

ingoing edges. The mechanics is in each case exactly the same: the graph Laplacian gathers differences between connected node values.

A corresponding expression can thus be written for the in-degree Laplacian, which maybe not surprisingly, involves the transpose:

$$\check{L}^{\alpha i}_{\beta j} = \check{D}^{\alpha i}_{\beta j} - A^{\alpha i}_{\beta j} \tag{64}$$

4.2 Properties of Directed Graph Laplacians

Graph Laplacian tensors re-express all the properties of the underlying graph. This means the adjacency tensor can be derived directly from the elements of either of the two graph Laplacians.

$$A^{\alpha i}_{\beta j} = \begin{cases} -\hat{L}^{\alpha i}_{\beta j} & : \text{ if } i \neq j \text{ and } \alpha \neq \beta \\ 0 & : \text{ otherwise} \end{cases} \tag{65}$$

Alas, the more flexible nature of directed graphs means many "nice" mathematical properties of undirected graphs are lost. For undirected graphs, the all-ones vector $\mathbf{1}$ is simultaneously a right and left eigenvector for the eigenvalue $\lambda = 0$ ($L\mathbf{1} = \mathbf{0}$ and $\mathbf{1}^T L = \mathbf{0}^T$). Whereas here, out-degree Laplacians only satisfy the condition that the sum over contravariant indices is zero.

$$\sum_{\beta j}^{mn^\beta} \hat{L}^{\beta j}_{\alpha i} = \sum_{\beta j}^{mn^\beta} \hat{D}_{\alpha i}\delta^{\beta j}_{\alpha i} - \sum_{\beta j}^{mn^\beta} A^{\beta j}_{\alpha i} \tag{66}$$

$$= \hat{D}_{\alpha i} - \hat{D}_{\alpha i} = 0 \tag{67}$$

and in-degree Laplacians only satisfy the corresponding condition that the sum of the covariant indices is zero.

In contrast with undirected graphs where the Laplacian is symmetric ($L^T = L$) for a directed graph it is generally asymmetric ($L^T \neq L$). This has significant repercussions: The undirected case is always positive semi-definite ($x^T L x \geq 0$), whereas the directed case is generally not. The undirected case thus admits an orthonormal basis of eigenvectors whereas in the directed case the eigenvectors are generally not orthogonal. In the undirected case all eigenvalues are real-valued and non-negative ($\lambda \in \mathbb{R}_{\geq 0}$) and can be ordered as $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, whereas in the directed case the eigenvalues are complex and admit fewer interpretations.

Nevertheless, while the in/out-degree graph Laplacians do not provide as insightful a view of the network, an artificial decomposition into symmetric and antisymmetric components does offer some new perspectives. First we transpose the graph Laplacian by swapping the entire source state (both node and layer) with the destination state:

$$\left(L^T\right)^{\alpha i}_{\beta j} = L^{\beta j}_{\alpha i} \tag{68}$$

Using transposition, the graph Laplacian uniquely splits into its symmetric and skew part. For example the out-Laplacian can be decomposed into:

$$\hat{L}^{\beta j}_{\alpha i} = \bar{L}^{\beta j}_{\alpha i} + \tilde{L}^{\beta j}_{\alpha i} \tag{69}$$

where the first part is a symmetric tensor $\bar{L}^{\beta j}_{\alpha i}$:

$$\bar{L}^{\beta j}_{\alpha i} = \frac{1}{2}\left(\hat{L}^{\beta j}_{\alpha i} + \hat{L}^{\alpha i}_{\beta j}\right) \tag{70}$$

and the second part is a antisymmetric tensor $\tilde{L}^{\beta j}_{\alpha i}$:

$$\tilde{L}^{\beta j}_{\alpha i} = \frac{1}{2}\left(\hat{L}^{\beta j}_{\alpha i} - \hat{L}^{\alpha i}_{\beta j}\right) \tag{71}$$

4.3 Models of Information Diffusion and Advection

> *"An Actor posts some content within their Community. The post strickes a chord and is reposted by a number of connections (specifically their following connections) within that Community. Some of them (possibly also the original Actor) might also be followed by Actors in other communities. In turn some of these Actors repost the content within their own communities etc. in an ever widening circle. At some point the interest and amplification of the post dies out and there is no further circulation of the content. It drifts to the bottom of the digital ocean, potentially never to be seen again..."*

Informally we might denote this all-too-familiar process of information propagating in social networks as the *diffusion* of information. The basic intuition when modeling diffusion phenomena on networks is that the Laplacian equilibrates high node values towards low valued nodes in the process diminishing the high node values. This is more or less directly following physical analogies in the diffusion of high concentrations of heat or chemical substances into a larger medium. Such diffusion may or may not capture accurately the nature of information propagation in actual social networks, but it is in any case a key modeling technique and building block, upon which one can construct more elaborate models.

When using Laplacian operators there is a *local interactions assumption* which implies that network dynamics and exchanges are intermediated only through *adjacency tensors that define the local neighborhood of each node*. This is less restrictive than it may sound, considering that edges can be established anywhere in the network! But this type of evolution is structurally different to, for example, the existence of a *global driving factor* that might influence all layers / nodes at the same time, irrespective of any network connectivity.

The standard procedure for building dynamic models on networks goes as follows: We imagine that a value at node (i, α) is diminished in proportion to its difference with other nodes (j, β) to which it is connected. We can write a general *evolution equation* over the network as follows:

$$\frac{d}{dt}X_{\alpha i} = -\sum_{\beta j}^{mn^{\beta}} \hat{L}^{j\beta}_{i\alpha} X_{\beta j} \tag{72}$$

$$= -\sum_{\beta j}^{mn^{\beta}} \bar{L}^{j\beta}_{i\alpha} X_{\beta j} - \sum_{\beta j}^{mn^{\beta}} \tilde{L}^{j\beta}_{i\alpha} X_{\beta j} \tag{73}$$

$$\tag{74}$$

where the graph Laplacian is decomposed into its symmetric (diffusive) and anti-symmetric ("advective") part[9].

[9] We put the advective part in quotes because a rank-4 tensor cannot really be represented as a pure vector field advection

4.4 Weighted and Normalized Graph Laplacians

For an edge-weighted directed graph described by a weights tensor W, the weighted Laplacians $\hat{L}, \check{L}$, can be defined entirely along similar lines:

$$\check{\mathcal{L}}^{\alpha i}_{\beta j} = \check{S}^{\alpha i}_{\beta j} - W^{\alpha i}_{\beta j} \tag{75}$$
$$\hat{\mathcal{L}}^{\alpha i}_{\beta j} = \hat{S}^{\alpha i}_{\beta j} - W^{\alpha i}_{\beta j} \tag{76}$$
$$\tag{77}$$

Incorporating both types of weights (nodes and edges) is accomplished by multiplying in element-wise fashion:

$$\check{\mathcal{L}}^{\alpha i}_{\beta j} = B^{\alpha i}(\check{S}^{\alpha i}_{\beta j} - W^{\alpha i}_{\beta j}) \tag{78}$$
$$\hat{\mathcal{L}}^{\alpha i}_{\beta j} = B^{\alpha i}(\hat{S}^{\alpha i}_{\beta j} - W^{\alpha i}_{\beta j}) \tag{79}$$
$$\tag{80}$$

Various scalings of Laplacians are useful for specific purposes and many variants have been introduced. One particularly noteworthy version is the *normalized* Laplacian used to analyze *Random Walks* on graphs. In this case the normalized version aims to scale all entries of the Laplacian by the corresponding out-degree values. We derive a normalized $\mathbb{L}^{\beta j}_{\alpha i}$ simply by factoring out the out-degree tensor:

$$\mathbb{L}^{\beta j}_{\alpha i} = \delta^{\beta j}_{\alpha i} - \frac{A^{\beta j}_{\alpha i}}{\hat{D}_{\alpha i}} \tag{81}$$

or explicitly:

$$\mathbb{L}^{\beta j}_{\alpha i} = \begin{cases} 1 & \text{if } i = j \text{ and } \alpha = \beta \\ -\frac{A^{\beta j}_{\alpha i}}{\hat{D}_{\alpha i}} & \text{if } i \neq j \text{ and there is a link between } (i, j) \\ 0 & \text{otherwise} \end{cases} \tag{82}$$

This Laplacian now takes the form $\mathbb{L} = I - P$, where P is a probability transition tensor:

$$P^{\beta j}_{\alpha i} = \frac{A^{\beta j}_{\alpha i}}{\hat{D}_{\alpha i}} \tag{83}$$

This tensor has positive elements (given both A and D are positive) and satisfies that the sum over contravariant indices is unity:

$$\sum_{\beta j}^{mn^\beta} P^{\beta j}_{\alpha i} = \sum_{\beta j}^{mn^\beta} \frac{A^{\beta j}_{\alpha i}}{\hat{D}_{\alpha i}} = 1 \tag{84}$$

References

[1] ActivityPub online documentation (as of jan 2026). Online Link.

[2] Bluesky Social, PBC. The Authenticated Transfer Protocol (AT Protocol) Specification, 2026. Online Link.

[3] Staschen, Björn and Foerster, Svenja and Ruthardt, Christian and Freihse, Christian. Decentralized Social Media Platforms as a Path to a More Resilient Information Ecosystem: Challenges Today and Recommendations for the Future. Report, Bertelsmann Stiftung, Gütersloh, Germany, January 2025.

[4] Leskovec, Jure and Lang, Kevin J. and Dasgupta, Anirban and Mahoney, Michael W. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[5] Kwak, Haewoon and Lee, Changhyun and Park, Hosung and Moon, Sue. What is Twitter, a Social Network or a News Media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 591–600, New York, NY, USA, 2010. ACM.

[6] Johan Ugander and Brian Karrer and Lars Backstrom and Cameron Marlow. The Anatomy of the Facebook Social Graph, 2011.

[7] Lucio La Cava, Domenico Mandaglio, Andrea Tagarelli. Polarization in decentralized online social networks. 2017.

[8] Lucio La Cava, Andrea Tagarelli. Information consumption and boundary spanning in decentralized online social networks: the case of mastodon users. *Online Social Networks and Media*, 30, 2022.

[9] Matteo Zignani, Sabrina Gaito, Gian Paolo Rossi. Follow the mastodon: Structure and evolution of a decentralized online social network. In *Proceedings of the Twelfth International AAAI Conference on Web and Social Media (ICWSM 2018)*, 2018.

[10] P. Papadopoulos. WP15: Connecting the Dots: Tensor Representations of ActivityPub Networks. *Open Risk White Papers*, 2024. Online Link.

[11] Newman, Mark. *Networks: An Introduction*. Oxford University Press, Oxford, UK, 2010.

[12] Manlio De Domenico et al. Mathematical formulation of multilayer networks. *Physical Review X*, 2013.

[13] M. Kivela et al. Multilayer networks. *Journal of Complex Networks*, 2014.

[14] S. Boccaletti et al. The structure and dynamics of multilayer networks. *Physics Reports*, 2014.

[15] Pixelfed Developers. FediDB: Fediverse Network Statistics. Online Link.

[16] Mastodon Documentation. Online Link.

[17] Activity Vocabulary online specification. Online Link.

[18] Activity Streams 2.0 online specification. Online Link.

[19] Bianconi, Ginestra. *Multilayer Networks: Structure and Function*. Oxford University Press, Oxford, UK, June 2018.

[20] Brauweiler, Michael and Pindiprolu, Meher Chaitanya and Pfeffer, Jürgen and Brandes, Ulrik. Decentralized Discourse: Interaction Dynamics on Mastodon. Websci '25, page 358–368, New York, NY, USA, 2025. Association for Computing Machinery.

[21] Watts, Duncan J. and Strogatz, Steven H. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[22] Weng, Lilian and Menczer, Filippo and Ahn, Yong-Yeol. Virality Prediction and Community Structure in Social Networks. *Scientific Reports*, 3(1):2522, Aug 2013.

[23] Gaël Guennebaud and Benoît Jacob and others. Eigen. Online Link.

[24] GraphBLAS Steering Committee. *GraphBLAS C API Specification*. GraphBLAS.org, version 2.1.0 edition, 2023. Online Link.

[25] P. Papadopoulos. WP10: Connecting the Dots: Concentration, diversity, inequality and sparsity in economic networks. *Open Risk White Papers*, 2021. Online Link.