Abstract—Research on graph representation learning has received great attention in recent years. However, most of the studies so far have focused on the embedding of single-layer graphs. The few studies dealing with the problem of representation learning of multilayer structures rely on the strong hypothesis that the inter-layer links are known, and this limits the range of possible applications. Here we propose MultiplexSAGE, a generalization of the GraphSAGE algorithm that allows embedding multiplex networks. We show that MultiplexSAGE is capable to reconstruct both the intra-layer and the inter-layer connectivity, outperforming competing methods. Next, through a comprehensive experimental analysis, we shed light also on the performance of the embedding, both in simple and multiplex networks, showing that both the density of the graph and the randomness of the links strongly influences the quality of the embedding.

Index Terms—Graph embedding, graph representation learning, link prediction, multiplex networks.

NOMENCLATURE

Symbol | Quantity
---|---
\(M\) | Multilayer network.
\(\mathcal{V}\) | Node set.
\(N\) | Number of nodes.
\(L\) | Number of layers.
\(\mathcal{V}_\alpha\) | Node set in layer \(\alpha\).
\(N_\alpha\) | Number of nodes in layer \(\alpha\).
\(E^\alpha\) | Edge set in layer \(\alpha\).
\(E^\text{intra}\) | Intra-layer link set.
\(E^\text{inter}\) | Inter-layer link set.
\(A^\alpha\) | Adjacency matrix of layer \(\alpha\).
\(C\) | Inter-layer adjacency matrix.
\(\mathcal{A}\) | Supra-adjacency matrix.
\(x_n \in \mathbb{R}^d\) | Input feature vector of node \(n\).

\(h_n^k \in \mathbb{R}^d\) | Representation vector of node \(n\) at step \(k\).
\(z_n \in \mathbb{R}^x\) | Output representation vector of node \(n\).
\(f_k\) | Aggregator function at step \(k\).
\(N^\mathcal{V}(n)\) | Node \(n\) intra-neighbor set.
\(N^\mathcal{V}_\alpha(n)\) | Node \(n\) inter-neighbor set.
\(\mathcal{P}(\mathcal{V})\) | Power set of set \(\mathcal{V}\).
\(W^\alpha_{kl} \in \mathbb{R}^{d_k \times d_{k-1}}\) | Inter-layer weight matrix at step \(k\).
\(W^\alpha_{\mathcal{V}\mathcal{V}} \in \mathbb{R}^{d_\mathcal{V} \times d_{\mathcal{V}-1}}\) | Self weight matrix at step \(k\).
\(\theta\) | Activation function.
\(J\) | Embedding loss function.
\(P(n)\) | Negative sampling probability distribution.
\(\sigma(x)\) | Sigmoid function.
\(\delta(L)\) | Inter-layer density parameter.
\(m_i\) | Number of inter-layer links among \(l\) layers.
\(\phi\) | Connection probability (Erdős-Rényi model).
\(\rho\) | Rewiring probability (Watts-Strogatz model).
\(Q\) | Number of neighbors (Watts-Strogatz model).
\(D\) | Network density.

I. INTRODUCTION

Graphs (or networks) are mathematical data structures to represent real-world complex systems. In the most general view, a graph consists of entities or objects (i.e., nodes), together with a set of relations (i.e., edges) between pairs of such entities. Examples of graphs include: social networks [1] in which entities are people and links represent friendship; co-authorship and citation networks [2]; biological networks [3] where nodes are for example proteins and relations between nodes may represent functional or physical interactions; knowledge graphs [4] (also known as semantic networks) in which nodes are real-world entities, such as objects, events, situations, and edges are relationships between them. In many contexts, simple graphs, where there is at most one edge between each pair of nodes are enough to model the application. However, graphs having different types of relations allow modeling, in a more comprehensive way, the complex system under study. Such a kind of graph is called multi-relational and is commonly divided into two important subgroups known as heterogeneous and multiplex graphs. In heterogeneous graphs, nodes have types, and therefore, we can partition the set of nodes into disjoint sets. Whereas, multiplex graphs can be decomposed in a set of \(k\) layers. Nodes can belong to different layers (each layer allows to
represent a specific kind of relation), along with intra-layer edge types for each layer. Furthermore, inter-layer edge types can exist, and these connect the same node across different layers.

Due to the ubiquity of networks in the real world, graph analysis has become mandatory in several domains [5]. In particular, the field of graph representation has received impressive research interest in the past few years [6], [7], [8]. Network representation learning or low dimensional network embedding consists of algorithmic methodologies, which allow projecting the nodes of a network into a multidimensional space preserving the structure of the network and its intrinsic properties. The usage of dimensionality reduction techniques to encode, into vectors, the high-dimensional information of nodes’ graph neighborhood allows then to apply traditional Machine Learning methodologies to networks. Indeed, such node embeddings can then be used in downstream learning tasks and analysis such as node classification [9], link prediction [10], and community detection [11].

The performance of machine learning methods crucially depends on the quality of the vector representations. Therefore, there is a wealth of research proposing a wide range of vector-embedding methods for various applications. Seminal works in this area include LINE [12], DeepWalk [13], and node2vec [14] which are based on taking short random walks in a graph and interpreting the sequence of nodes seen on such random walks as if they were words appearing together in a sentence. However, from a deep learning perspective, such embedding methods are all shallow, indeed they directly optimize the output vectors without hidden layers. Although the shallow approach can be generalized to multi-relational graphs it presents several limitations. Shallow embedding methods are transductive. Indeed, these methods are only able to generate embeddings for nodes that were present during the training phase.

To go beyond such limitations, shallow encoders can be replaced with more sophisticated encoders that take into account the structure and attributes of the graph. Among the most popular encoders, we have graph neural networks (GNNs). The key feature of a GNN is that it uses a kind of neural message passing, which essentially is a generalization of the Weisfeiler-Lehman test [15], [16]. At each iteration, every node aggregates information from its local neighborhood. As these iterations go on, each node embedding contains more information obtained from remote nodes of the graph. In the last few years, research in this area has been proposed in this context [7], [8], [17], [18], [19]. One of the most prominent examples of an inductive node-embedding tool based on GNNs is GraphSAGE [20].

More recently, research models for multi-layer network embedding have risen. In [21] authors propose an embedding approach for multigraphs, in [22] a generalization of deep-walk and node2vec for multiplex graphs was introduced. In [30] the authors propose a Multi-Relational Graph Convolutional Network (MR-GCN) based on a novel convolution operator on multi-relational graphs. In particular, MR-GCN is established on the eigen decomposition defined by the generalized tensor product. No embedding of inter-layer link is provided since networks are treated as multi-view, i.e., multigraphs. Other relevant approaches include [24], [25], [26], [27], [28]. However, in all of these approaches, no distinction between inter-layer and intra-layer connections is taken into account. Furthermore, such embedding models that make use of the concept of multi-view networks are based on the assumption that each layer has exactly the same nodes, therefore there is no need to predict inter-layer links. In the most general case, in a multiplex network we could be unaware of the presence of the same nodes in each layer. E.g., in systems biology, in a multiplex network linking genotypic and phenotypic information of disease to discover novel disease-disease relations, different kinds of nodes and inter-layer links between the two layers can be present [29]. In [23] the authors propose a semi-supervised task in which the aim is to develop a robust tensor-based deep semi-supervised learning (SSL) approach over multi-relational graphs. Tensor graph convolutional networks (TGCN) maps each node $n$ to its label $y_n$; and hence, it infers the unavailable labels resulting in a supervised approach. Authors propose two successive phases of intra-neighborhood aggregation and inter-neighborhood aggregation, furthermore the inter-layer aggregation is many-to-many (all inter-layer links exist), and therefore inter-layer link prediction is not necessary.

In this article, we introduce MultiplexSAGE, a generalization of the GraphSAGE algorithm for embedding multiplex networks. Our embedding approach focuses only on unlabeled graphs but can be easily generalized to labeled graphs. Through a comprehensive experimental analysis based on three benchmark datasets, we shed light also on the limit of the vector embedding in producing reliable predictions, both for simple graphs and multiplex. Indeed, our analysis shows that both density of the graphs and the randomness of the links actually limits the quality of the embedding.

II. MULTIPLEX NETWORKS

A multiplex network is a generalization of a network that allows to naturally represent systems of connected units when multiple types of interactions among them exist [31], [32]. Multiplex networks can be seen as a particular type of multilayer networks [33], [34]. We will first define them by describing their graph and matrix representation, and presenting the crucial concept of intra-layer and inter-layer links, as well as the one of supra-adjacency matrix [35].

Let us consider a set $V$ of $N$ nodes interacting through $L$ different types of relations, represented as $L$ different layers. Let us assume that $N_{\alpha}$ nodes lie on each layer $\alpha$, with
α = 1, ..., L so that $N_1 + N_2 + \cdots + N_L = N$. Each layer of the multiplex consists in a network $G_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$, where $\mathcal{V}_\alpha \subseteq \mathcal{V}$ is the set of $N_\alpha$ nodes in the layer, while $\mathcal{E}_\alpha$ is a set of edges representing the relations of type $\alpha$. Hereby, we will refer to the edges in $\mathcal{E}_{\text{intra}} = \bigcup_{\alpha=1}^L \mathcal{E}_\alpha$ as intra-layer links. The connectivity of each layer $G_\alpha$ can be encoded as an adjacency matrix $A^\alpha \in \mathbb{R}^{N_\alpha \times N_\alpha}$, whose element $a_{ij}^\alpha$ is equal to one, if $(i, j) \in \mathcal{E}_\alpha$, i.e., nodes $i$ and $j$ interacts through the relation $\alpha$, while it is equal to zero otherwise.

In addition to the intra-layer connectivity, we assume to connect some of the nodes lying on different layers, so that given $\alpha, \beta$, with $\alpha \neq \beta$, a node lying on layer $\alpha$ can be linked with at most one node on layer $\beta$. This constraint makes the multiplex a suitable representation for those systems where the same set of individuals are connected through different relations [34]. In what follows, we will call these connections inter-layer links. As a further hypothesis, we assume that if a node $i$ on layer $\alpha$ is connected to a node $j$ on layer $\beta$, and if $j$ is connected to a node $k$ on layer $\gamma$, then nodes $i$ and $k$ are also connected. Therefore, the graph $G_{\text{inter}} = (\mathcal{V}, \mathcal{E}_{\text{inter}})$, where $\mathcal{E}_{\text{inter}}$ is the set of inter-layer links, will be formed by disconnected components that are either cliques or isolated nodes [35]. Such a graph can be as well characterized by an adjacency matrix $C \in \mathbb{R}^{N \times N}$, whose element $c_{ij}$ is equal to one, if $(i, j) \in \mathcal{E}_{\text{inter}}$, whereas it is equal to zero otherwise.

Finally, we can define the supra-adjacency matrix $\tilde{A}$, which encodes both intra-layer and inter-layer connections. By labeling the nodes according to the labels of the layer, i.e., indices from 1 to $N_\alpha$ denote the nodes in the first layer, from $N_1 + 1$ to $N_1 + N_2$ the nodes in the second layer, and so on, we can write the supra-adjacency matrix as

$$\tilde{A} = \bigoplus_{\alpha} A^\alpha + C \quad (1)$$

where $\bigoplus$ denotes the direct sum. An example of a multiplex network and its supra-adjacency matrix is shown in Fig. 1. Note that the diagonal blocks correspond to the adjacency matrices of the graphs at each layer, thus encoding the intra-layer connectivity, while the off diagonal blocks contain the information on the inter-layer connections. It is worth mentioning that we could interpret $\tilde{A}$ as the adjacency matrix of a single-layer network $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E} = \mathcal{E}_{\text{intra}} \cup \mathcal{E}_{\text{inter}}$. However, it is crucial to remark the differences between multiplexes and single-layer networks. In multiplexes there are two types of links, i.e., intra-layer and inter-layer, representing two distinct kinds of connections. While intra-layer links model (multiple kind of) relations between different individuals, inter-layer links represent the fact that two individuals in different networks of interactions are actually the same individual. Such a difference is lost in single-layer networks, where all links are of the same type. This is also reflected in the structure of the supra-adjacency matrix. In facts, $\tilde{A}$ is a block matrix, where each block corresponds either to the connectivity within a single layer, or to the links between different layers. As for single-layer networks all links are equivalent, the corresponding adjacency matrix does not have a clear block partition. Multiplexes are also different from multigraphs. Though multigraphs, as multiplexes, admit multiple types of interactions, they are single-layer graphs, so they don’t have inter-layer connections. Also, in principle in multiplex networks each layer could consist of a multirelational graph. Therefore, as the goal of this article is to develop an embedding algorithm that deals with intra-layer and inter-layer links as two distinct entities, we rely on a multiplex representation.

Let us remark that our definition is more flexible compared to the one usually given for multiplex networks. Indeed, it is common to assume that each layer of the multiplex is composed of the same number of nodes, i.e., $N_\alpha = m, \forall \alpha$, and that for any $\alpha, \beta$, a node on layer $\alpha$ is connected to exactly one node on layer $\beta$ [32], [34]. Here, to extend the range of possible applications of our algorithm, we do not consider such extra constraints. As an example, let us consider the case of different online social networks (OSNs), which can be represented by a single entity, namely a multiplex network where each layer encodes the connections in one of the OSNs. If a user can have only one account for each OSN, we constraint ourselves to the case of a multiplex. In such an example, assuming that each layer is composed of the same number of nodes and that all the possible inter-layer links exist is too restrictive. First, the users might not have an account on every OSN. Second, these accounts might not be connected one another, i.e., it is not known a priori that two accounts on different OSNs are owned by the same user. Hence, relaxing the constraint of having all the possible inter-layer connections opens to relevant applications like, for instance, the one of de-anonymization [36], which can be seen as an inter-layer link prediction.

III. MULTIPLEXSAGE ALGORITHM

The embedding algorithm MultiplexSAGE represents the natural generalization of the GraphSAGE algorithm [20] to multiplex networks. In order to provide a vector representation $z_n$ for a graph node $n$, GraphSAGE relies on the idea of training a set of $K$ aggregator functions $f_k$, with $k \in \{1, \ldots, K\}$, that learn to aggregate information from the node $K$th neighborhood. In other words, given a node $n$, its embedding vector depends, according to a first aggregator function, on the features of its first-neighbors, which in turn are evaluated, through a second aggregator function, from the characteristics of their first-neighbors, i.e., the second-neighborhood of $n$, and so on, up to a certain depth $K$. Therefore, the vector $z_n$ can be determined once all the aggregator functions are trained.

In the original version of the GraphSAGE algorithm, which deals with the embedding of (single-layer) graphs, when considering a specific node $n$, each aggregator $f_k$ is a function of the features of the node itself and of the ones of its first-neighborhood. Conversely, since for multiplex networks we can distinguish, for each node, both an intra-layer and an inter-layer neighborhood, in the MultiplexSAGE embedding algorithm the aggregator functions will depend on the features of node $n$, and of its intra-neighbors, but also on the features of its inter-neighbors. The MultiplexSAGE embedding generation is formally described by Algorithm 1.

Again, the core idea behind the embedding algorithm is to generate, for each node of the multiplex network, a vector representation by aggregating the features of the node $K$th neighborhood, through a series of $K$ iterations. This time, however, at each iteration $k \in \{1, \ldots, K\}$, each node $n$ is
Algorithm 1 MultiplexSAGE Embedding Generation Algorithm

Input: Multiplex network $\mathcal{M} = (\mathcal{V}, L, c_{\text{intra}}, c_{\text{inter}})$; input feature vector $\mathbf{x}_n, \forall n \in \{1, \ldots, |\mathcal{V}|\}$; depth $K$; differentiable aggregator functions $f_k, \forall k \in \{1, \ldots, K\}$; intra-layer and inter-layer neighborhood functions $\mathcal{N}_H, \mathcal{N}_V : \mathcal{V} \rightarrow \mathcal{P}(\mathcal{V})$

Output: Vector representation $\mathbf{h}_n, \forall n \in \{1, \ldots, |\mathcal{V}|\}$

for $k = 1, \ldots, K$ do

for $n = 1, \ldots, |\mathcal{V}|$ do

$h_n^k \leftarrow f_k([h_m^{k-1}, \forall m \in \mathcal{N}_H(n)], [h_m^{k-1}, \forall m \in \mathcal{N}_V(n)], h_n^{k-1})$

end for

end for

$\mathbf{z}_n \leftarrow h_n^K, \forall n \in \{1, \ldots, |\mathcal{V}|\}$

TABLE I

| Network | Nodes | Layers | Intra-layer links | Inter-layer links |
|---------|-------|--------|------------------|------------------|
| arXiv   | 19310 | 13     | 45657            | 20738            |
| Drosophila | 11867 | 7      | 40228            | 5173             |
| IF-ww-yt | 11827 | 3      | 74815            | 6028             |

$k - 1$ of its inter-neighbors, and the representation vector at step $k - 1$ of the node itself, all of which are multiplied by the respective weight matrices. Given this choice for the aggregator functions, once the algorithm is provided with the activation function $\theta$, the only unknown variables that have to be determined through the training process are the weight matrices $W_H^k, W_V^k$ and $S^k$.

Consistently with the GraphSAGE algorithm, to learn the weight matrices defining the MultiplexSAGE aggregator functions in an unsupervised setting, we introduce a loss function based on the supra-adjacency matrix of the multiplex network. In particular, the loss function $J_{\mathcal{M}}$ is defined to force neighboring nodes (both intra-layer and inter-layer) to have similar embedding vectors, while constraining nodes that are not connected, i.e., the negative links, to have dissimilar representations. However, since the number of negative links can be high, the computation of the loss function for large networks can be computationally expensive. To reduce the computational cost of the learning process, it is common to rely on negative sampling [37], [38], namely to evaluate the loss function only on a random subset of all the possible negative links. Formally, given the embedding vectors $\mathbf{z}_n, \forall n \in \mathcal{V}$, we define

$$ J_{\mathcal{M}} = -\sum_{(n,m) \in \mathcal{E}} \log(\sigma(\mathbf{z}_n^T \mathbf{z}_m)) + \sum_{m \sim P(n)} \log(\sigma(-\mathbf{z}_n^T \mathbf{z}_m)) $$

where $P(n)$ is the negative sampling probability distribution, and $\sigma(x)$ is the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$. Note that the sum in the curl brackets has $Q$ terms, corresponding to the number of negative samples.

IV. DATA AND EVALUATION

A. Multiplex Network Data

To test the performance of the MultiplexSAGE algorithm in predicting both intra-layer and inter-layer connections, we have analyzed different types of real-world multiplex networks, such as collaboration, biological, and OSNs, as well as synthetic datasets. For each empirical dataset, we consider only the largest connected component, and we ignore, if these are given, the directions and the weights of the links, i.e., we assume all the networks to be undirected and unweighted.

1) arXiv [39]: The arXiv multiplex network is a collaboration network where each layer represents a different category, i.e., research topic, of the pre-print archive. To generate the network, only the articles including the word “networks” in the title or in the abstract published before May 2014 were selected. The network largest connected component consists in this...
case of 19,310 nodes over 13 layers, connected through 48,657 intra-layer connections, and 20,738 inter-layer ones.

2) **Drosophila** [40], [41]: This is the protein-genetic interaction multiplex network of the common fruit fly *Drosophila melanogaster*, where layers correspond to interactions of different nature. The dataset is gathered from the biological general repository for interaction datasets (BioGRID), updated to January 2014. The largest connected component consists of 118,670 nodes over seven layers, with 40,228 intra-layer links, and 5,713 inter-layer ones.

3) **ff-tw-yt** [42]: This is a multiplex network obtained from Friendfeed (ff), a social media aggregator, which allows the users to register their accounts on other OSNs. The retrieved multiplex network consists of users who registered a single Twitter (tw) account and a single YouTube (yt) account on ff, and whose tw and yt accounts are associated with one ff account. The largest connected component consists of 11,287 nodes over three layers, with 74,815 intra-layer connections, and 6,028 inter-layer ones.

The main characteristics of the datasets considered in this study are reported in Table I. It is worth remarking that, for all the multiplex networks described above, no external feature vectors are provided for the nodes. Therefore, for each node $n$ we will consider as the input feature vector $x_n$ a one-hot encoder vector [43], i.e., $x_n,i = \delta_{n,i}$, where $\delta$ is the Kronecker delta. Hence, node $n = 1$ will be represented by the vector $x_1 = [1, 0, 0, \ldots, 0]^T$, node $n = 2$ by $x_2 = [0, 1, 0, \ldots, 0]^T$, and so on, up to node $n = N$, represented by vector $x_N = [0, \ldots, 0, 1]^T$.

### B. Experimental Setup and Evaluation

The main task we want to test MultiplexSAGE on is that of predicting both the intra-layer and the inter-layer links of a multiplex network. To assess the performance of our algorithm, we consider the following experimental setup. We first select a random sample of 20% of the network nodes, which we refer to as marked nodes. We then define the test and the training sets. Each one of these sets consists of positive examples, i.e., the links of the network, and negative examples, i.e., couples of unconnected nodes. As positive examples in the test set, we consider a subset of 20% of the intra-layer links and all the inter-layer links of the marked nodes. Positive examples of the training set consist of all the remaining intra-layer and inter-layer links of the entire network. As concerns the negative examples, we include in the test set 20% of all the possible negative intra-layer links among the marked nodes, and all the possible inter-layer links among them. The remaining uncoupled pair of nodes in the entire network form the negative examples of the training set.

Since the algorithm is trained so that adjacent nodes have a similar vector representation, while nonadjacent nodes have a dissimilar one, we would expect MultiplexSAGE to discriminate between positive and negative links in terms of the similarity of their vertices. Indeed, when it performs well, the similarity between the vertices of a positive link should be higher than the similarity between the vertices of a negative link. Instead of setting an arbitrary discrimination threshold on the value of the vertex similarity, so that all links whose vertex similarity is above the threshold are considered true links of the networks and vice versa, to estimate the algorithm performance we rely on the receiving operating characteristic (ROC) curves. To construct them, for both intra-layer and inter-layer links, we evaluate the frequency distribution of the vertex similarity for positive and negative links, respectively, and estimate how the true positive rate, namely the fraction of positive links correctly predicted as true links, and the false positive rate, i.e., the fraction of negative links incorrectly predicted, vary as a function of the discrimination threshold. Therefore, as a scalar measure of the algorithm performance, we consider the area under the ROC curve (AUC), which corresponds to the probability that a randomly chosen positive link has a vertex similarity higher than one of a randomly chosen negative link. Therefore, when the algorithm is able to distinguish between positive and negative links, the value of the AUC tends to 1, while in the opposite case it goes to 0.5, i.e., random classifier.

Instead of performing one single experiment, on each dataset considered, we run the embedding procedure multiple times, each time taking a different set of marked nodes, hence different training and test sets. As a measure of the algorithm performance, we take the average AUC, using the standard deviation as a measure of the statistical error.

### C. Competing Methods

We compare MultiplexSAGE to two competing methods, namely GraphSAGE and general attributed multiplex heterogeneous network embedding (GATNE). In what follows we provide their main features.

1) **GraphSAGE** [20]: GraphSAGE is an algorithm for inductive node embedding, which leverages node features to learn an embedding function that generalizes to unseen nodes. It allows to simultaneously learn the topological structure and the distribution of node features of a node neighborhood. As GraphSAGE is developed for single-layer network embeddings, we train the algorithm making no distinction between intra-layer and inter-layer links.

2) **GATNE** [21]: GATNE is an embedding algorithm for attributed multiplex heterogeneous networks, i.e., multigraphs with different types of nodes and edges. To adapt GATNE to our task, we define two types of edges, namely intra-layer and inter-layer links, thus producing a multigraph which the algorithm can learn to embed.

Other methods, such as those proposed in [22], [24], [25], [26], [27], [30], and [23], rely on the hypothesis that each layer has exactly the same nodes and, consequently, that the inter-layer connections among them are known. Hence, since inter-layer link prediction is the main application we are interested in, we cannot use those algorithms as a benchmark for MultiplexSAGE.

### V. Results

In this section, we illustrate the potential of MultiplexSAGE. First, we compare the performance of our algorithm with that of the competing methods in predicting the intra-layer and
inter-layer connectivity of the multiplex networks described above. Then, we analyze the dependence of MultiplexSAGE and GraphSAGE on the number of layers of a network, assessing how the performances vary as we gradually change the number of layers to embed. Finally, we will further investigate such a dependence, showing how the prediction accuracy of GraphSAGE is related to the link density and to the randomness of the network.

### A. Embedding Multiplex Networks

Our first analysis consists in evaluating the performance of MultiplexSAGE in predicting intra-layer and inter-layer links when embedding a multiplex network. To do so, for each of the three datasets illustrated, we consider the experimental setup described in Section IV-B, training and testing the algorithm on 20 different samplings of the marked nodes and of the train and test sets. As a measure of the algorithm performance, we consider the AUC averaged over the different realizations of the embedding procedure. We use the standard deviation as a measure of the error. As a benchmark, we compare the results obtained by MultiplexSAGE with the one of GraphSAGE and GATNE.

Table II illustrates the results obtained by the algorithms, on the three datasets, for both intra-layer and inter-layer link prediction. First, we note that, for both GraphSAGE and MultiplexSAGE, the value of the average AUC tends to be generally higher for the inter-layer link prediction than for the intra-layer link prediction, while the opposite is true for GATNE. On all the multiplex networks considered, MultiplexSAGE provides significantly better predictions of the inter-layer connectivity compared to both GraphSAGE and GATNE, as confirmed by Welch’s t-test (for the three datasets considered, namely ff-tw-yt, Drosophila, and arXiv, respectively, p-values are $p = 2.2 \cdot 10^{-9}$, $p = 6.0 \cdot 10^{-22}$, and $p = 3.3 \cdot 10^{-36}$, for the comparison with GraphSAGE, and $p = 2.0 \cdot 10^{-5}$, $p = 5.0 \cdot 10^{-19}$, and $p = 2.4 \cdot 10^{-21}$, for the comparison with GATNE). When reconstructing the intra-layer connections, the MultiplexSAGE and GraphSAGE have similar performances, while GATNE has the best performance. Indeed, the differences between GraphSAGE and MultiplexSAGE in the AUC for the ff-tw-yt and the arXiv datasets are not statistically significant ($p = 0.058$) and ($p = 0.23$), respectively, while the out-performance of MultiplexSAGE by GraphSAGE is significant for the Drosophila dataset ($p = 4.3 \cdot 10^{-7}$). Instead, GATNE significantly outperforms MultiplexSAGE in the intra-layer link prediction (p-values are $p = 1.6 \cdot 10^{-12}$, $p = 2.4 \cdot 10^{-26}$, and $p = 2.0 \cdot 10^{-30}$, for ff-tw-yt, Drosophila, and arXiv datasets, respectively). It is important to remark, however, that the main application of MultiplexSAGE is inter-layer link prediction, a task on which our algorithm performs significantly better than both GraphSAGE and GATNE, for all datasets considered.

Compared to GraphSAGE, our embedding algorithm seems to display a sort of compensation effect, so that, to enhance its performance in the reconstruction of inter-layer connections, it loses accuracy when predicting the intra-layers links. However, it is worth remarking that, while the loss in the intra-layer prediction accuracy is low (5.9% for the Drosophila, and 1.4% for the arXiv datasets), the gain in the inter-layer prediction accuracy is substantial (22.2% and 18.6%, respectively). This result utterly motivates the usage of MultiplexSAGE as a tool for inter-layer link prediction.

We remark that the performance of all algorithms seems to depend positively on the number of layers of the network. Indeed, when predicting the intra-layer links, both MultiplexSAGE and GraphSAGE obtain results that are comparable to those of a random classifier, i.e., AUC = 0.5, on the ff-tw-yt network, which has three layers, and on the Drosophila network, composed by seven layers. Conversely, on the arXiv network, which is formed by thirteen layers, the value of the AUC goes up to about 0.70 for both algorithms. As regards the inter-layer link prediction, the dependency on the number of layers seems to be more pronounced, with the value of AUC ranging from 0.62 and 0.56 for the OSN, to 0.83 and 0.70 for the collaboration network, for MultiplexSAGE and GraphSAGE respectively. The performance of GATNE in the inter-layer link prediction also depends on the number of layers, as the algorithm is able to outperform a random classifier only on the arXiv network. Nonetheless, the performance in the intra-layer link prediction seems not to be correlated with the number of layers in the multiplex network.

Overall, for both intra-layer and inter-layer link prediction, the number of layers of the network seems to have a positive impact on the performance of the algorithms, so the more the number of layers the higher the value of the AUC. In Section V-B, we will provide an in-depth analysis of the dependence of the algorithm performance on the number of layers of the network, introducing an explanatory measure for the accuracy of the link prediction.

### B. Varying the Number of Layers

As we have shown in Section IV-A, the accuracy of MultiplexSAGE in intra-layer and inter-layer link prediction seems to depend on the number of layers of the network. To further analyze this dependence, here we carry on the following study. For the Drosophila and the arXiv datasets, we first sort the $L$ layers according to their number of nodes, so that $N_1 > N_2 > \cdots > N_L$. Then, we construct a set of multiplex sub-networks, the first being formed by layers $[1, 2]$, i.e., the two layers with the largest number of nodes, the second formed by layers $[1, 2, 3]$, and so on. Finally, the last network of the set consists in the original multiplex network, formed by all the $L$ layers. Therefore, for each network in the set, we consider the experimental setup previously described, again training and testing the algorithm over 20 different sampling of the marked nodes. For this analysis, we focus on MultiplexSAGE and GraphSAGE, measuring their accuracy by adopting the AUC averaged over multiple realizations of the embedding procedure.

Fig. 2 shows the results obtained for Fig. 2(A) Drosophila and Fig. 2(B) arXiv multiplex networks. First, one can observe that, for both datasets, the accuracy of MultiplexSAGE in the prediction of the inter-layer links is generally higher compared to the one of GraphSAGE, for every sub-network considered. The only exception consists of the embedding of
TABLE II

| Algorithm   | fi-tw-yt intra | fi-tw-yt inter | Drosophila intra | Drosophila inter | arXiv intra | arXiv inter |
|-------------|----------------|----------------|------------------|-------------------|-------------|-------------|
| GraphSAGE   | 0.47 ± 0.02    | 0.56 ± 0.02    | 0.54 ± 0.02      | 0.63 ± 0.02      | 0.72 ± 0.02 | 0.70 ± 0.01 |
| GATNE       | 0.83 ± 0.01    | 0.47 ± 0.01    | 0.78 ± 0.01      | 0.55 ± 0.01      | 0.91 ± 0.01 | 0.63 ± 0.01 |
| MultiplexSAGE | 0.48 ± 0.02    | 0.62 ± 0.02    | 0.51 ± 0.01      | 0.77 ± 0.02      | 0.71 ± 0.02 | 0.83 ± 0.01 |

Fig. 2. Variation of the AUC for inter-layer (blue lines) and the intra-layer (orange lines) link prediction as a function of the number of layers, for MultiplexSAGE (solid lines) and GraphSAGE (dashed lines), on (A) Drosophila and (B) arXiv multiplex networks.

Fig. 3. Variation of the AUC for inter-layer link prediction (green line) and of the parameter δ (orange line) as a function of the number of layers, on (A) Drosophila and (B) arXiv multiplex networks.

We now want to further investigate the trend of the AUC curves for the inter-layer link prediction, aiming at explaining why the accuracy depends on the number of layers. To do so, let us define the following parameter:

$$
\delta(L) = 1 - \frac{m_L}{\sum_{l=2}^{L}(l-1)N_l} \in [0, 1]
$$

where $m_L$ is the number of inter-layer links when $L$ layers are considered, and $N_l$ are the number of nodes at each layer $l$. Intuitively, this parameter corresponds to the density of inter-layer links that are not present in the network. To clarify this, let us consider $L = 2$, i.e., a two-layer network. In this case, the parameter $\delta$ is simply given by

$$
\delta(2) = 1 - \frac{m_2}{N_2}
$$

where $m_2$ is the number of inter-layer nodes between the two layers and $N_2$ is the number of nodes in the smallest layer.
i.e., \(N_2 \leq N_1\). When all the nodes in layer 2 are connected to a node on layer 1 we have \(m_2 = N_2\), thus \(\delta(2) = 0\). However, if there are nodes on layer 2 that do not have an inter-layer neighbor on layer 1, one has \(m_2 < N_2\), which leads to \(\delta(2) > 0\), up to the case where no inter-layer links exist, for which \(\delta(2) = 1\). Note that the term \((l - 1)\) within the summation comes from the fact that, when we add the \(l\)-th layer to the network, we could connect each of the \(N_l\) nodes to \((l - 1)\) nodes, one for each of the \((l - 1)\) layers already present.

Fig. 3 shows the comparison between the AUC curves for the inter-layer link prediction and the parameter \(\delta(L)\) defined above, for Fig. 3(A) Drosophila and Fig. 3(B) arXiv multiplex networks. As one can note, for both datasets \(\delta(L)\) well correlates with the AUC curve. Such a result suggests that MultiplexSAGE, at least when performing a featureless network embedding, better predicts the inter-layer connectivity when the density of inter-layer links is low, i.e., when \(\delta\) is closer to one. To have an intuition of the role of the link density in the algorithm accuracy, let us imagine to perform the embedding of a (almost) complete graph using GraphSAGE. As (almost) all the nodes in the network are adjacent, at each step of the embedding procedure they will be characterized by the same vector representation. As a consequence, the accuracy of the link prediction will be low, since all the connections in the test set, both positive and negative, will be predicted as positive. In Section V-C, we will further investigate such a result, showing that the GraphSAGE (and consequently the MultiplexSAGE) accuracy is affected not only by the network sparsity but also by its randomness so that the more the graph is random the worse the performance of the algorithm is.

### C. Impact of the Network Randomness

As we have discussed in Section V-B, the accuracy of the inter-layer link prediction in the MultiplexSAGE algorithm seems to depend on the density of connections. To better understand such a result, here we come back to GraphSAGE, which shares the same embedding strategy as MultiplexSAGE, while being a simpler algorithm to examine. We hereby perform two different numerical analyses. Initially, we study how the accuracy of the link prediction changes as we randomly add links to a network, i.e., as we make the network denser.

Then, we investigate the impact on GraphSAGE performance of randomly rewiring the network links, namely, we explore how making the network more random affects the link prediction accuracy.

We begin by further analyzing the role of link density in prediction accuracy. To carry out the study, we generate a set of networks constructed by starting from the largest layer of the arXiv dataset and adding a fraction of links \(\rho\) between nonadjacent nodes. As the additional links are homogeneously distributed, we can think the newly constructed graphs as the union\(^1\) between the original graph and an Erdös-Rényi (ER) random graph [44], with \(N_1\) nodes and connection probability \(\rho\). Considering also the starting network (\(\rho = 0\)), we account for a total of eleven networks, that we embed following the usual experimental setup, training and testing the algorithm over 20 different samplings of the marked nodes. Again, to measure the prediction accuracy, we consider the AUC averaged over the different runs of the embedding procedure.

Fig. 4 displays how the average AUC varies as a function of the fraction of additional links \(\rho\). In agreement with the results of Section V-B, when the network is sparse, i.e., for \(\rho < 10^{-4}\), GraphSAGE is able to distinguish between positive and negative links, as shown by the high value of the AUC, which is close to 0.75. However, when the density of links increases, the accuracy of the algorithm starts to decrease, down to AUC \(= 0.5\) (\(\rho \approx 10^{-3}\)), where GraphSAGE performs as a random classifier. We can therefore conclude that, when embedding a graph with featureless nodes, GraphSAGE is able to accurately reconstruct the network connectivity only when the network itself is sparse, whereas it fails to do so for graphs that are dense.

We now want to analyze how network randomness influences the accuracy of the algorithm predictions. To do so, we construct a collection of networks using the Watts and Strogatz (WS) model [45]. In particular, starting from a ring lattice of \(N = 10^4\) nodes, where each node is connected to \(Q = 4\) neighbors, two on each side, we generate the networks by rewiring the links of the lattice with a certain probability

\(^1\)The union of two graphs \(G_1 = (V_1, E_1)\) and \(G_2 = (V_2, E_2)\) is here defined as the graph \(G = (V, E) = (V_1 \cup V_2, E_1 \cup E_2)\). Note that, as in the case here considered we have \(V_1 = V_2\), it follows that \(V = V_1 = V_2\).
φ. One crucial feature of the WS model is that we can tune the network from a regular structure (φ = 0) to a disordered one (φ = 1) while keeping the link density constant, i.e., $D = Q/(N - 1)$. Therefore, the WS model allows studying how the randomness of the network connectivity affects the accuracy of the link prediction, leaving out the contribution of edge density. Including also the regular lattice (φ = 0), we consider eleven networks in total. Again, we embed each network following the experimental setup described above, with 20 different train-test splits. As an accuracy measure, we once again evaluate the AUC averaged over the different realizations of the embedding procedure.

Fig. 5 shows the variation of the average AUC as a function of the rewiring probability φ. In particular, we observe that the accuracy of GraphSAGE in the link prediction is negatively affected by the randomness of the network. Indeed, for smaller rewiring probabilities, i.e., φ < 10^{-2}, the AUC is over 0.85, meaning that the algorithm is performing well in the positive and negative links discrimination. However, for higher values of φ, the accuracy of the models starts to decrease. In particular, for φ = 1 GraphSAGE almost behaves as a random classifier, as AUC = 0.55. Summing up, this analysis shows that randomness can negatively affect the performance of the algorithm. In particular, the more a network is similar to a homogeneous random graph the worse will be the prediction accuracy. On the other hand, the results obtained suggest that networks with a regular structure or with a high clustering coefficient, such as the one generated by the WS model for small values of the rewiring probability, allow for better performances in the link prediction task. Finally, it is worth mentioning once again that such an analysis is performed in a condition where no node is provided with an external feature vector. We leave such a study for future work.

VI. CONCLUSION AND DISCUSSION

In this article, we have introduced an algorithm for the embedding of multiplex networks, which distinguishes between intra-layer and inter-layer edges and produces reliable results, especially for the prediction of inter-layer links. We have tested the algorithm performance on different types of empirical multiplex networks. The results of our analysis clearly show that taking into account the multi-layer nature of the network positively influences the quality of the embedding. On the other hand, we also found that both increasing the density of links and shuffling the network edges negatively influences the quality of the embedding.

Our results pave the way for further research in different directions. Future works may focus on extending the analysis of MultiplexSAGE by exploring different aggregator functions, such as LSTM and pooling aggregators [20]. Indeed, the two latter aggregators rely on the idea of computing a set of weight matrices, so they could be applied to the case of multilayer networks by considering two distinct sets of matrices, one for the intra-layer neighborhood and one for the inter-layer neighborhood.

Here we considered the case of undirected and unweighted networks, which is the one accounted by GraphSAGE. However, a further direction to investigate would be the extension of our algorithm to the case of both directed and weighted networks [46]. Also, an adaptation of MultiplexSAGE to deal with sparse multiplex networks would represent an important extension of this work [47].

The development of further embedding algorithms that are specific to multiplex networks is also a fundamental future direction, given the various applications for which these structures are used. In particular, future works should focus on elaborating GNN algorithms for multilayer network embedding, as little attention has been given so far to the topic.

Lastly, a critical analysis of the interplay between the network structure and the performance of the embedding is fundamental to the research field. In this article, we have focused on the role of the density and randomness of links, but other topological features, such as node clustering, community structures, or the presence of hubs, may have a profound impact on the quality of the embedding.

ACKNOWLEDGMENT

Luca Gallo would like to thank C. Monti for the useful discussion and for his precious suggestions on how to perform the numerical analysis.

REFERENCES

[1] J. Scott, “Social network analysis,” Sociology, vol. 22, no. 1, pp. 109–127, 1988.
[2] X. Liu, J. Bollen, M. L. Nelson, and H. Van de Sompel, “Co-authorship networks in the digital library research community,” Inf. Process. Manage., vol. 41, no. 6, pp. 1462–1480, Dec. 2005.
[3] A.-L. Barabasi, N. Gulbahce, and J. Loscalzo, “Network medicine: A network-based approach to human disease,” Nature Rev. Genet., vol. 12, no. 1, pp. 56–68, Jan. 2011.
[4] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, “A survey on knowledge graphs: Representation, acquisition, and applications,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 2, pp. 494–514, Feb. 2022.
[5] J. Zhou et al., “Graph neural networks: A review of methods and applications,” AI Open, vol. 1, pp. 57–81, Jan. 2020.
[6] D. Bacciu, F. Errica, A. Micheli, and M. Podda, “A gentle introduction to deep learning for graphs,” Neural Netw., vol. 129, pp. 203–221, Sep. 2020.
[7] F. Chen, Y.-C. Wang, B. Wang, and C.-C.-J. Kuo, “Graph representation learning: A survey,” APSIPA Trans. Signal Inf. Process., vol. 9, no. 1, pp. 1–15, 2020.
[8] M. Grohe, “word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data,” in Proc. 39th ACM SIGMOD-SIGACT-SIGAI Symp. Princ. Database Syst., Jun. 2020, pp. 1–16.
[9] Y. Rong, W. Huang, T. Xu, and J. Huang, “DropEdge: Towards deep graph convolutional networks on node classification,” 2019, arXiv:1907.10903.
[10] L. Lu and T. Zhou, “Link prediction in complex networks: A survey,” Phys. A, Stat. Mech. Appl., vol. 390, no. 6, pp. 1150–1170, Mar. 2011.
[11] S. Fortunato, “Community detection in graphs,” Phys. Rep., vol. 486, nos. 3–5, pp. 75–174, Feb. 2010.
[12] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “LINE: Large-scale information network embedding,” in Proc. 24th Int. Conf. World Wide Web, 2015, pp. 1067–1077.
[13] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2014, pp. 701–710.
[14] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” inProc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 855–864.
[15] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, “Weisfeiler-Lehman graph kernels,” J. Mach. Learn. Res., vol. 12, no. 9, pp. 1–23, 2011.
[16] A. Leman and B. Weisfeiler, “A reduction of a graph to a canonical form and an algebra arising during this reduction,” Nauchno-Technicheskaia Informatsia, vol. 2, no. 9, pp. 12–16, 1968.

[17] M. Liu, H. Gao, and S. Ji, “Towards deeper graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2020, pp. 338–348.

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” 2017, arXiv:1710.10903.

[19] H. Gao and S. Ji, “Graph U-nets,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 9, pp. 4948–4960, Sep. 2022.

[20] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” 2017, arXiv:1706.02216.

[21] Y. Cen, X. Zhou, Z. Zhang, H. Yang, J. Zhou, and J. Tang, “Representation learning for attributed multiplex heterogeneous network,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 1358–1368.

[22] M. Gong, W. Liu, Y. Xie, Z. Tang, and M. Xu, “Heuristic 3D interactive walk for multilayer network embedding,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 7, pp. 3309–3321, Sep. 2020.

[23] Z. Huang, X. Li, Y. Ye, and M. K. Ng, “MR-GCN: Multi-relational graph convolutional networks based on generalized tensor product,” in Proc. 29th Int. Joint Conf. Artif. Intell., Jul. 2020, pp. 1258–1264.

[24] W. Liu, P. Chen, S. Yeung, T. Suzumura, and L. Chen, “Principled multilayer network embedding,” in Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW), Nov. 2017, pp. 134–141.

[25] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, “An attention-based collaboration framework for multi-view network representation learning,” in Proc. ACM Conf. Inf. Knowl. Manage., Nov. 2017, pp. 1767–1776.

[26] H. Zhang, L. Qiu, L. Yi, and Y. Song, “Scalable multiplex network embedding,” in Proc. 27th Int. Joint Conf. Artif. Intell., Jul. 2018, pp. 3082–3088.

[27] Y. Shi et al., “Mvn2vec: Preservation and collaboration in multi-view network embedding,” 2018, arXiv:1801.06597.

[28] C. Yang et al., “MultiSage: Empowering GCN with contextualized multi-embeddings on web-scale multipartite networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2020, pp. 2434–2443.

[29] A. Halu, M. De Domenico, A. Arenas, and A. Sharma, “The multiplex network of human diseases,” npj Syst. Biol. Appl., vol. 5, no. 1, p. 15, Apr. 2019.

[30] V. N. Ioannidis, A. G. Marques, and G. B. Giannakis, “Tensor graph convolutional networks for multi-relational and robust learning,” IEEE Trans. Signal Process., vol. 68, pp. 6535–6546, 2020.

[31] M. De Domenico et al., “Mathematical formulation of multilayer networks,” Phys. Rev. X, vol. 3, no. 4, Dec. 2013, Art. no. 041022.

[32] F. Battiston, V. Nicosia, and V. Latora, “Structural measures for multiplex networks,” Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 89, no. 3, Mar. 2014, Art. no. 032804.

[33] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, “Multilayer networks.” J. Complex Netw., vol. 2, no. 3, pp. 203–271, Sep. 2014.

[34] S. Boccaletti et al., “The structure and dynamics of multilayer networks,” Phys. Rep., vol. 544, no. 1, pp. 1–122, Nov. 2014.

[35] E. Cozzo, G. F. De Arruda, F. A. Rodrigues, and Y. Moreno, Multiplex Networks: Basic Formalism and Structural Properties. Springer, 2018, doi: 10.1007/978-3-319-92255-3.

[36] S. Ji, P. Mittal, and R. Beyah, “Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey,” IEEE Commun. Surveys Tuts., vol. 19, no. 2, pp. 1305–1326, 2nd Quart., 2017.

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 3111–3119.

[38] Z. Yang, M. Ding, C. Zhou, H. Yang, J. Zhou, and J. Tang, “Understanding negative sampling in graph representation learning,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2020, pp. 1666–1676.

[39] M. De Domenico, A. Lancichinetti, A. Arenas, and M. Rosvall, “Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems,” Phys. Rev. X, vol. 5, no. 1, Mar. 2015, Art. no. 011027.

[40] M. De Domenico, V. Nicosia, A. Arenas, and V. Latora, “Structural reducibility of multilayer networks,” Nature Commun., vol. 6, no. 1, pp. 1–9, Apr. 2015.

[41] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers, “BioGRID: A general repository for interaction datasets,” Nucleic Acids Res., vol. 34, no. 90001, pp. D535–D539, Jan. 2006.

[42] M. E. Dickison, M. Magnani, and L. Rossi, Multilayer Social Networks. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[43] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, arXiv:1609.02907.

[44] P. Erdös and A. Rényi, “On random graphs I,” Pub. Math. (Debrecen), vol. 44, no. 90001, pp. D535–D539, Jan. 2006.

[45] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” Nature, vol. 393, no. 6684, pp. 440–442, Jun. 1998.

[46] X. Luo, H. Wu, Z. Wang, J. Wang, and D. Meng, “A novel approach to large-scale dynamically weighted directed network representation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 12, pp. 9756–9773, Dec. 2022.

[47] X. Luo, Y. Yuan, S. Chen, N. Zeng, and Z. Wang, “Position-transitional particle swarm optimization-incorporated latent factor analysis,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 8, pp. 3958–3970, Aug. 2022.

Lucia Gallo received the Ph.D. degree in complex systems for physical, socio-economic and life sciences from the University of Catania, Catania, Italy, in 2022. He is currently a Post-Doctoral Fellow with Central European University, Vienna, Austria. His research interests include dynamical systems, complex networks, and higher-order networks.

Vito Latora received the Ph.D. degree in theoretical physics from the University of Catania, Catania, Italy, in 1996. He is currently a Professor of applied mathematics, the Chair of complex systems, and the Head of the Complex Systems and Networks Research Group, School of Mathematical Sciences, Queen Mary University, London, U.K. He is also a Professor of physics with the University of Catania. He is the coauthor of more than 200 scientific publications. His research interests include structure and dynamics of multiplex networks, temporal networks, and higher-order networks. He is interested in applications of network science to study innovation ecosystems and to understand the social and neural aspects of human creativity. Dr. Latora is an Editor of the Journal of Complex Networks.

Alfredo Pulvirenti received the Ph.D. degree in computer science from the University of Catania, Catania, in 2003. He is currently an Associate Professor of computer science with the University of Catania. He is the coauthor of more than 140 scientific publications. His research interests mainly focus on bioinformatics and data analysis with applications in biomedicine. In particular, he studies methods for pathway and biological network analysis, drug repositioning, eRNAs, design of pipelines for RNA-seq and DNA-Seq data, subgraph matching, and motif finding.