Generalized Euclidean Measure to Estimate Distances on Multilayer Networks

MICHELE COSCIA, IT University of Copenhagen, Denmark

Estimating the distance covered by a spreading event on a network can lead to a better understanding of epidemics, economic growth, and human behavior. There are many methods solving this problem – which has been called Node Vector Distance (NVD) – for single layer networks. However, many phenomena are better represented by multilayer networks: networks in which nodes can connect in qualitatively different ways. In this paper, we extend the literature by proposing an algorithm solving NVD for multilayer networks. We do so by adapting the Mahalanobis distance, incorporating the graph's topology via the pseudoinverse of its Laplacian. Since this is a proper generalization of the Euclidean distance in a complex space defined by the topology of the graph, and that it works on multilayer networks, we call our measure the Multi Layer Generalized Euclidean (MLGE). In our experiments, we show that MLGE is intuitive, theoretically simpler than the alternatives, performs well in recovering infection parameters, and it is useful in specific case studies. MLGE requires solving a special case of the effective resistance on the graph, which has a high time complexity. However, this needs to be done only once per network. In the experiments, we show that MLGE can cache its most computationally-heavy parts, allowing it to solve hundreds of NVD problems on the same network with little to no additional runtime. MLGE is provided as a free open source tool, along with the data and the code necessary to replicate our results.

CCS Concepts: • **Information systems → Data mining**.

Additional Key Words and Phrases: complex networks, social networks, network distance, propagation

ACM Reference Format:
Michele Coscia. 2022. Generalized Euclidean Measure to Estimate Distances on Multilayer Networks. *ACM Trans. Knowl. Discov. Data.* 1, 1, Article 1 (January 2022), 22 pages. https://doi.org/10.1145/3529396

1 INTRODUCTION

Complex networks are a powerful tool that allows us to investigate a number of questions about many different complex phenomena: how epidemics spread [6], how countries [21, 22] and producers in general [33, 34] explore the space of new profitable activities, and how people adopt new behaviors [19] or products [27] are but few examples among the many. All these research trails share a common format: they represent a complex space (social or economic) as a network and investigate how an event propagates on them. This propagation is usually some sort of process activating/deactivating the nodes of the network. One way to represent this mathematically is by connecting the status of each node to a number – e.g. how many times the node performed a given function. We call "node vector" the data structure storing these values.

One crucial part of the answer to the questions posed in these papers relies on estimating how far the process spreads in the network. This is a non-trivial quantity to estimate, because it involves calculating a (weighted) distance between groups of nodes – or, to better say, the distance between

Author's address: Michele Coscia, mcos@itu.dk, IT University of Copenhagen, Rued Langgaards Vej 7, Copenhagen, Denmark, 2300.

1556-4681/2022/1-ART1 $15.00
https://doi.org/10.1145/3529396

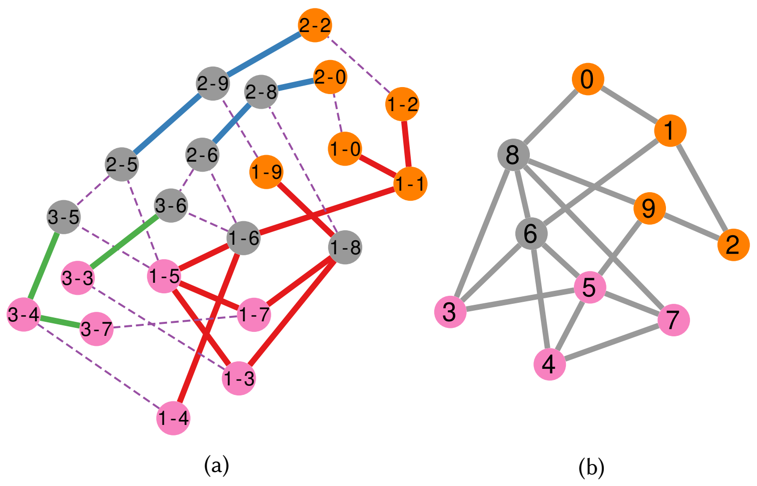


Fig. 1. Two examples of NVD scenarios. (a) A multilayer network. Solid edges are intra-layer relationships, whose color represent the layer. Dashed purple edges are inter-layer couplings connecting a node with its identities in the other layer. Node labels identify each identity as "layer id–node id". Orange and pink nodes are active in two different moments in time, gray nodes are never active. (b) The single layer view of (a), collapsing all nodes connected by a dashed purple edge and all edges between pairs of collapsed nodes. A node is orange (or pink) if any of its multilayer identities was orange (or pink) in (a).

the two node vectors, which needs to take into account the network topology conecting these nodes. There are many ways to do so, as summarized in a recent survey [10], which labeled this problem as the Node Vector Distance (NVD) problem. However, there is a crucial limitation in this literature: so far, all methods solving NVD do so on networks with a single layer. None of these methods work on multilayer networks.

In this paper, we propose a method that can be used for multilayer networks. In a multilayer network, like in Figure 1(a), nodes can connect via different qualitative types of relations [3, 24] – e.g. friendship, enmity, financial, leisure, collaboration, etc. In this paper, we aim at answering the question: what is the distance between orange and pink nodes in Figure 1(a)? How do different layers (differently colored solid edges) influence this distance calculation? And how do we count jumping between layers, by following the dashed purple edges connecting different identities of the same entity?

The latter two are important questions, because multilayer networks are a sophisticated tool that is able to represent more complex phenomena than single layer networks. Multilayer networks have proven their usefulness in showing that particular properties of complex networks only emerge when considering all the layers [4], proving that a true multilayer analysis is more powerful than the sum of single layer analysis on each of the layers separately.

Moreover, methods that are agnostic of the different layers through which nodes connect could result in incorrect estimates: e.g., assuming social relationships exert influence on people's behavior, but ignoring the difference between a friendship and an enmity link. For instance, Figure 1(b) shows a version of Figure 1(a) that one would construct if they were unaware about multiplexity. Is the distance between the orange and the pink nodes in Figure 1(b) an accurate approximation of the real distances from Figure 1(a)? Would the answer be the same if we knew that the blue and green layers actively obstruct the propagation?

This paper is an extension of previous work in which we defined the Generalized Euclidean (GE) NVD solution [7]. In GE, one estimates the distance between the status of nodes at time t with their status at time $t + 1$ by calculating the effective resistance between the two statuses. This is achieved by calculating the pseudoinverse of the graph Laplacian, solving the heat equation in a discrete setting, with the graph constraining the underlying topology. In this paper, we formulate a way to create a multiplex graph Laplacian, leading to a Multi Layer Generalized Euclidean (MLGE). We can plug this multiplex Laplacian in the equation to solve the problem for multilayer networks. Our formulation is flexible, because it allows us to specify different propensities for different layers to conduct the spreading process, and also to define how likely is the process to jump across any two layers of the network.

In our experiments, we show that MLGE has an intuitive behavior and it is theoretically simpler than other possible multilayer adaptations of NVD-solving algorithms – specifically, the multilayer versions of the Earth Mover Distance (MLEMD) and of the Graph Fourier Transform (MLGFT). MLGE also performs well in recovering infection parameters on a number of real world networks. We show MLGE's usefulness in a case study, highlighting the structural importance of airlines connecting European countries.

MLGE is not particularly computationally efficient, because it needs to calculate the pseudoinverse of the Laplacian. However, under the assumption that an analyst is interested in simulating thousands of spreading events on the same network, its structure allows for caching of the most computationally-heavy parts. As a result, the time it takes to run a single spread event is roughly indistinguishable from running thousands of them, a feature that, e.g., MLEMD does not have.

The code and the data necessary to replicate the results in the experiment section of this paper are freely available[1]. We release MLGE as an open source tool free for use. The code can be obtained by downloading the same archive.

2 RELATED WORK

This paper provides a contribution to solve the NVD problem. A survey about methods dedicated to these solutions has recently been published [10]. As we will make clear in our problem definition (Section 3), NVD aims at estimating the distance between two vectors on a space defined by a graph. NVD, in the sense of "network distance", should not be confused with similarly named problems. Examples are graph-graph distance [26, 36], where we calculate the distance between two graphs instead of one (and no vectors); or graph-node vector correlation [20], where we estimate the likelihood that a node vector has spread through the edges of a graph; or graph variance [12], where one measures how spread out in a graph the values of a vector are. In the latter two cases, there is only one node vector, not two as in NVD.

Among the NVD methods discussed in the literature, we decide to extend the Generalized Euclidean method [7] to incorporate multilayer network data. Two alternatives to GE are the Earth Mover Distance [18] and the Graph Fourier transform [45, 46]. Since we use these two methods as the comparison with our own method, we explain them more in details in Section 4.4. We only note that EMD has been studied for a long time in the literature, and that there are plenty of approaches to perform the optimization step that it requires [14, 15, 29, 38, 39]. Each of these optimizations could be conceivably used to create a sub-class of NVD solutions based on the EMD idea.

There are many other classical computer science problems that could conceivably be adapted to solve NVD. One is Multi-Agent Path Finding (MAPF). MAPF is roughly equivalent to EMD: we have a certain number of robots moving on a graph and we want to move them from an origin configuration to a target configuration. The difference is that we impose a capacity on nodes and

[1]https://www.michelecoscia.com/?page_id=1733#nvdml

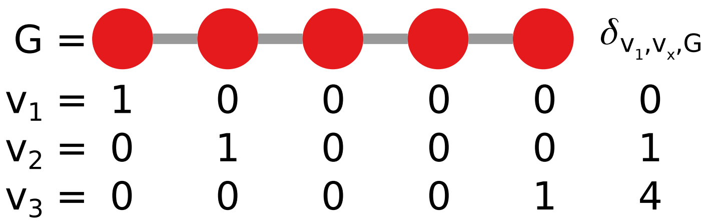


Fig. 2. An example of simple NVD. Each value in v_x corresponds to the node in G above and the final column reports a simple δ distance between each vector and v_1.

edges, specifically no robot can use the same node or edge at the same time with another robot. Different scenarios and optimizations can generate their own NVD measure [1, 28, 30].

Another classical problem is Discrete Pursue Evasion. This is like MAPF with two teams of robots: the pursuers and the evaders. Once a pursuer meets an evader in a node, they annihilate each other. The time it takes for total annihilation could be used to estimate the NVD between pursuers and evaders. Again, different scenarios and different optimization strategies can create a new NVD subclass [32, 47].

All EMD, MAPF, and DPE variants are barely used to solve NVD and, when they are, it is exclusively in a single layer scenario. Thus, our contribution of a multilayer NVD is unique in the literature.

Solving NVD in multilayer networks is useful. NVD has many applications. For instance, NVD could be useful to discover the β contagion factor in network epidemics [6] (as we do in Section 5.4), which is also useful for evaluating the success of viral marketing campaign [27] (replacing pathogens with product adoptions); estimating how dynamic is the export basket of a country [21, 22], or the industrial composition of a region's industry space [34]; calculating the similarity of two images for computer vision algorithms [40, 42]; and performing signal processing tasks such as frequency analysis [43] and trend filtering [48]. Finally, NVD can be used to calculate network correlations of node attributes [9]. All these applications benefit from representing the data as a multilayer network, and thus would need a multilayer NVD measure.

3 PROBLEM DEFINITION

3.1 Formulation

In NVD we have three inputs. We have two node vectors (v_1 and v_2) and a multilayer graph G. We want to find a distance function $\delta_{v_1,v_2,G}$ such that: $v_1 \times v_2 \times G \mapsto \mathbb{R}_0^+$, i.e. $\delta_{v_1,v_2,G}$ takes the inputs and returns a real positive number. The output tells us how far v_1 is from v_2, constrained by the complex space defined by G.

Figure 2 shows a simplified example to aid intuition. Here, we count as $\delta_{v_1,v_2,G}$ the number of edges between the node in one vector and the node in the other. This simple $\delta_{v_1,v_2,G}$ would break down if we have a more complex topology in G, or if we have more than one non-zero value in v_x.

Since δ is a distance estimation it must be zero or positive, it must be zero if we calculate the distance between v_1 and itself, and it must be reflective (i.e. the distance between v_1 and v_2 is equal to the distance between v_2 and v_1). Triangle inequality is not a necessary property for $\delta_{v_1,v_2,G}$, since there are many distance measures that do not have this property (e.g cosine distance). The measure we define in this paper does respect triangle inequality, but this is not necessary.

We now provide more details about the inputs (G in particular), since their shape is what determines the main contribution of this paper.

Differently from all other NVD papers in the literature, we represent G as a labeled graph $G = (V, E, D, C)$. Here, V is the set of nodes, E is the set of edges, D is the set of layers/dimensions and C is the set of inter-layer couplings. Each edge in E is a quadruple (u_1, u_2, d, w), with $u_1, u_2 \in V$; $d \in D$; and $w \in \mathbb{R}$ is a real-valued edge weight.

The couplings in C are special edges. They are a triple (u_1, u_2, w), and they are semantically different from edges in E. An edge in E represents a relation between nodes u_1 and u_2 of type d. In this case u_1 and u_2 are distinct entities. A coupling in C represents an identity across layers between u_1 and u_2, meaning that they refer to the *same* entity in *different* layers. Going back to Figure 1(a), the connection between nodes 3-4 and 3-7 is an edge in the green layer (layer 3) connecting two different entities (entities 4 and 7). The connection between nodes 3-4 and 1-4 is a coupling connecting two nodes belonging to the same entities (entity 4) across two layers (layers 1 and 3).

What the layers represent in the real world is normally problem-dependent: some problems require to split a network into layers, while others do not. In general, a network has layers if it contains connections that are qualitatively different from each other and/or switching back and forth between different ways of connecting two nodes requires some extra effort.

For instance, in a social network, we can consider a friendship to be qualitatively different from a work collaboration, because they come with different dynamics and expectations – although, if these distinctions are irrelevant for the problem at hand, they can be compressed into a generic "social relationship". In this case, the inter-layer coupling connects a person with all its personas: crossing an inter-layer coupling means that the person is switching between different modes of thinking: "am I relating with this person as a friend or as a co-worker?"

In another example, two airlines connecting the same airports in a flight network can be considered as different layers because, for travelers, switching between planes of different companies comes at a certain monetary and practical costs (e.g. switching between terminals) – even if one might ignore this technicality if they want to talk exclusively about which airports can be reached from which others, no matter the airline. In this case, the inter-layer coupling tells us we are in the same physical space – the airport – and crossing it reports the cost a traveler incurs when switching airlines to continue their trip.

Note that, in this formulation, edge weights are *capacities*, not *costs* [8]. A capacity-type weight tells you how related two nodes are. For instance, if you are analyzing a road network, they will tell you the throughput of a trait of road in number of cars (or lanes). A cost-type weight tells you how hard it is to pass through the edge. In the same example, this could be the length of the trait of road (in kilometers or in the time it is required to pass through it). This is an important distinction to make when comparing our approach with the alternatives (in Section 4.4).

If the data does not have information about relationship strength, all weights can be set to the same constant and will not affect the result.

The input vectors v_1, v_2 must have length $|V|$. Each vector v_x determines the activation weight of each node of G. Entries in v_x must be zero or positive real values. If an entry is zero, we say that the node is inactive. A non-zero value determines how strongly the node is active. If the data only contains data about the active/inactive status, v_x can be a binary vector. One could also normalize v_x so that it sums to one.

3.2 Motivation

We need G to be multilayer because many complex systems are better represented by mutlilayer networks, and events propagating on multilayer structures might not be representable in single layer approximations. We provide two examples.

First, one might avoid the multilayer problem by analyzing a multilayer network one layer at a time, aggregating the results. This would not work for NVD. One reason why is that a layer in

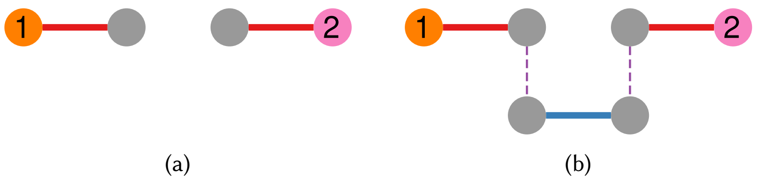


Fig. 3. The node color represents the occupancy in v_1 (orange) and v_2 (pink). The edge color represents the layer. The purple dashed lines are inter-layer couplings.

isolation might be disconnected, as for instance in Figure 3(a). In this case, one cannot calculate a propagation distance between nodes 1 and 2, because disconnected components cannot allow propagation between them. However, in presence of multiple layers (Figure 3(b)), the propagation can happen via a different layer – in blue in the figure.

Second, one could solve the previous issue by simply collapsing all layers into a single layer network. However, by doing so, we lose the information contained in the couplings C, which might speed up or delay the spread.

4 METHODOLOGY

Here, we propose to solve multilayer NVD by extending the existing Generalized Euclidean (GE) measure to a Multi Layer Generalized Euclidean (MLGE). We start by defining the simple Generalized Euclidean measure as done in the original paper [7] (Section 4.1), and then we move on to increasingly complex settings: unweighted multilayer (Section 4.2) and weighted multilayer (Section 4.3).

4.1 Single Layer

The core issue in this paper is estimating the distance between two vectors, v_1 and v_2. The most obvious choice is assuming that the vectors live in an n-dimensional Euclidean space. The number of dimensions is the length of the vector which, in our case, is the number of nodes in the network: $|V|$. Then, one can simply calculate the Euclidean distance between the two points identified by the vectors: $\delta_{v_1,v_2} = \sqrt{(v_1 - v_2)^T(v_1 - v_2)}$. In this formula, $(v_1 - v_2)$ is the element-wise difference of the vectors, while $(v_1 - v_2)^T$ is its transpose.

The problem with the Euclidean distance is that each dimension contributes equally to the spatial distance between the points. In a network, this is not the case. Since each dimension is a node in the network, some dimensions contribute less to the distance than others. If two vectors only differ along two dimensions, it makes a difference whether the two corresponding nodes are connected or not.

The Mahalanobis distance solves the problem of differential contribution to the total distance by different dimensions. In the Mahalanobis distance, we multiply the squared vector difference by the inverse of the vectors' covariance matrix S: $\delta_{v_1,v_2} = \sqrt{(v_1 - v_2)^T S^{-1}(v_1 - v_2)}$. The interpretation is that some dimensions are correlated with each other and thus contain less unique information than others. Therefore, each of them should contribute less to the overall distance. S only depends on the vectors we are comparing, thus it also ignores G's topology, as does the Euclidean distance.

In this paper we propose to replace the covariance matrix S with a matrix Q which contains the graph's topological information. One constraint we have to respect is that Q needs to be positive (semi)definite, otherwise the $x^T Qx$ product could be negative for some vector x, which would result in a nonsensical distance estimation. For this reason, we cannot use the adjacency matrix of the graph, which is not positive semidefinite – unless the graph is empty.

We focus on the graph Laplacian L due to its relationship with diffusion processes [5] and its use in solving the effective resistance calculation between pairs of nodes in a graph [25]. In practice, one can estimate the distance between two nodes in a graph by measuring the difference in electric potential they would have if the graph were an electric circuit [13]. This is more advantageous than using the shortest path distance, because the shortest path distance is heavily affected by the addition/removal of a handful of edges, while the effective resistance is a more robust measure [12].

Discussing how to estimate the effective resistance between two nodes goes beyond the scope of this paper, but the relevant portion here is that it involves the calculation of the Moore-Penrose pseudoinversion of the Laplacian (L^+), which contains an estimation of the distance between the nodes in the network. The Laplacian L is the result of the difference between the degree and the adjacency matrices of G, with the degree matrix being a matrix with the degrees of the nodes on the main diagonal and zeros elsewhere.

We need to calculate the Moore-Penrose pseudoinverse because L is singular and thus it cannot be inverted. The pseudoinversion requires solving the singular value decomposition problem, i.e. finding the matrix Σ such that $Q_1 \Sigma Q_2^T = L$. Then one can calculate Σ^+, which contains the reciprocals of Σ and satisfies the equation $Q_2 \Sigma^+ Q_1^T = L^+$. L^+ is now the pseudoinverse of L, since it holds that $LL^+L = L$ and $L^+LL^+ = L^+$.

L^+ is a proper candidate for our Q matrix, defining a proper metric between v_1 and v_2 over G as:

$$\delta_{v_1,v_2,G} = \sqrt{(v_1 - v_2)L^+(v_1 - v_2)}.$$

4.2 Unweighted Multilayer

The Generalized Euclidean (GE) measure derived in the previous section only considers L as originating from an unweighted and single layer G. Here, we need to generalize L to the weighted multilayer case. Here we start by redefining how L is calculated for unweighted multilayer networks.

As a starting point, we calculate L not by using the degree and the adjacency matrix of G, but by using B: the oriented incidence matrix of G.

B is an $|V| \times |E|$ matrix. In the column of edge (u_1, u_2), there is one 1 in the row corresponding to one vertex and one -1 in the row corresponding to the other vertex. All other rows have 0. Thus all columns of B sum to zero. Note that it does not matter which one between u_1 and u_2 gets the 1 and which one gets the -1, as long as the columns sum to zero. This matters for directed graphs showing how our method could also be applied to directed multilayer graphs, even if we ignore them in this paper.

One can calculate the graph Laplacian of an unweighted single layer G as: $L = BIB^T$, where I is the identity matrix.

If all inter-layer couplings in a multilayer network have the same strength and this strength is equal to the intra-layer edges in E, then one could simply make B as a $|V| \times (|E| + |C|)$ matrix, and obtain the multilayer Laplacian accordingly. Note that one would still have to decide the style of the multilayer coupling. Usually, this would be a clique style, where all nodes belonging to the same entity are connected to each other (Figure 4(a)). However, alternatives exists, e.g. determining a layer order and then connecting the nodes belonging to the same entity with a chain (Figure 4(b)). Layers can be coupled in an arbitrary graph, as long as all the different identities of the same entity are part of the same connected component.

Since now we have a proper L, we can apply the classical GE formula to this simple unweighted multilayer network. L is a $|V| \times |V|$ matrix, whose -1 entries are either edges or couplings. Its main diagonal is the sum of the degree of the node plus all its couplings. Thus, this simplified

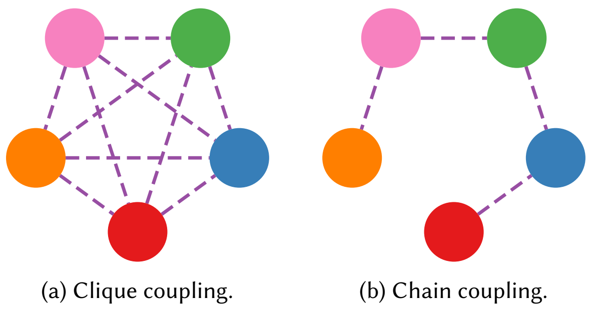


Fig. 4. The node color represents the layer to which it belongs. The purple dashed lines are inter-layer couplings.

L is equivalent to the Laplacian of the supra-adjacency matrix [41]. Section 4.3 details how to distinguish edges from couplings. The order in which dimensions D are considered is not important.

4.3 Weighted Multilayer

One can create a weighted Laplacian by replacing the identity matrix I with a properly constructed weight matrix W. In our case, W must be a diagonal $(|E| + |C|) \times (|E| + |C|)$ matrix. Each entry in the diagonal represents the weight of a specific edge or inter-layer coupling. If the data already contains edge weights and inter-coupling strengths, this can be passed directly to generate the graph Laplacian as $L = BWB^T$.

In absence of data about edge and coupling weights, one might still want to differentiate between the propagation happening inside a layer and the one happening across layers. This requires two (sets of) parameters:

(1) Interlayer jump cost parameter(s): how easy it is to jump across layers. This could be a $|D| \times |D|$ layer-layer matrix, if each layer pair has a different relation, or it could be a single parameter if moving across any two layers is equivalent. This defines the C portion of W.
(2) Intralayer cost(s): how much it costs to propagate inside a specific layer d. This is a layer vector of length $|D|$ if each layer has a different cost. This defines the E portion of W.

In both cases, but most likely for the couplings C, one might want to specify an infinite edge weight. Inter-layer couplings in multilayer networks often connect different identities of the same entity. This would imply that, if something affects an entity, it affects all of its identities at the same time. As a consequence, the coupling strength is infinite. Our method can handle this scenario by collapsing nodes connected by infinite-strength couplings into the same node – and thus simplifying B by reducing its dimensions.

When collapsing nodes across layers, one has to be aware that edges might need to be aggregated as well. If you collapse node 1 into node 2 and node 3 into node 4, if both the $(1, 3)$ and the $(2, 4)$ edges existed in different layers, they would collapse into the same edge. Thus we also need to specify a function aggregating the weights of these edges. For MLGE, it makes sense to sum edge weights: since edge weights are capacities, the new capacity connecting the collapsed nodes is the sum of the old capacities connecting the different identities of the nodes. However, this is not the only reasonable choice: as we will see in Section 5.2, measures based on shortest paths only use the maximum weight between the two collapsed ones, thus we have to aggregate edge weights by taking the maximum rather than the sum.

Now that the Laplacian $L = BWB^T$ includes both weights and multilayer couplings, we can solve the multilayer NVD problem by applying the classical GE formula.

L has the same shape as in the unweighted multilayer case described in Section 4.2, with the difference that now its entries are weighted and there is a way to quantify the difference between edges and couplings.

4.4 Alternative Approaches

There are two alternatives to solve NVD: Graph Fourier Transform (GFT) and the Earth Mover Distance (EMD).

4.4.1 Graph Fourier Transform. In GFT, one filters the input vectors v_x such that their entries take into account the connections between the nodes they represent [45]. The idea is that each value in v_x is the result of the combination of a true signal and the correlations between sensors (the nodes). Just like in signal processing, the GFT can tease those correlation out to reconstruct the true signal. The two filtered input vectors can then be compared directly.

The filter is achieved by calculating the eigenvectors of the Laplacian of G. If $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_n$ are the sorted eigenvalues of L and $l_0, l_1, \ldots, l_n$ are the corresponding eigenvectors, then $\Phi = (l_0, l_1, \ldots, l_n)$ and Λ is the diagonal matrix with the eigenvalues on the diagonal and zeroes everywhere else. The filter of v is then achieved as:

$$\hat{v} = \Lambda \Phi^T v.$$

GFT was never developed with multilayer networks in mind but, given that it rests on the Laplacian just like MLGE, we can develop an MLGFT by constructing the weighted multilayer Laplacian in the same way we outlined in the previous sections.

4.4.2 Earth Mover Distance. In EMD, one assumes that the objective is to transport the weights from the v_1 occupancy to v_2. We want to do so with the minimal number possible of edge crossing [18]. This is not limited to graphs, but is valid for any two points u_1, u_2 in a space with a properly defined distance δ_{u_1,u_2} that respects triangle inequality. If M is the set of moves, the objective is:

$$M = \underset{m_{u_1,u_2}}{\arg\min} \sum_{u_1} \sum_{u_2} m_{u_1,u_2} \delta_{u_1,u_2}.$$

m_{u_1,u_2} tells us the amount of weight that we want to transport from u_1 to u_2. The EMD distance is then simply the sum of M, i.e. the sum of distances between all u_1s and u_2s, weighted by how much weight we can transfer between them. In a graph, one can use the shortest path distance between u_1 and u_2 as δ_{u_1,u_2}. While working on multilayer networks, to get an MLEMD measure one has to make sure to take into account the layer switching cost, which can be encoded in the weight of the coupling edges in C.

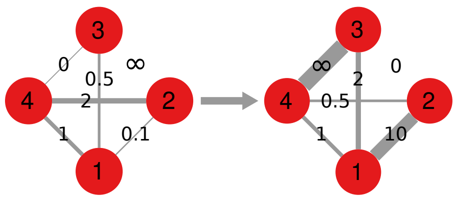
Minimizing M is not trivial. We use the Pele and Werman [38, 39] implementation[2].

4.4.3 Theoretical Comparison. In Section 5 we provide a few empirical examples of when MLGE is superior to MLGFT and MLEMD. Here, instead, we briefly mention some minor theoretical advantages over the alternatives.

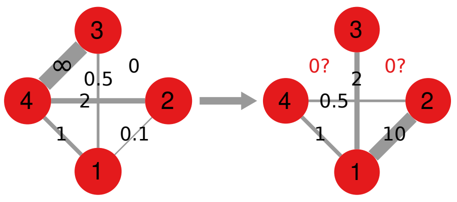
As mentioned in Section 3.1, MLGE interprets the edge weights as capacities. MLGFT and MLEMD, instead, see weights as costs. This has both theoretical and practical advantages for MLGE.

If edge weights are costs, it is difficult to represent instantaneous transitions. This would imply to have an edge weight of zero, which would be confused with an absent edge. The resulting adjacency matrix would have two different "types" of zeroes, with different semantics. There is, in principle, no issue in having an infinite edge weight, provided this case is handled as we do in Section 4.3.

[2]https://github.com/wmayner/pyemd



(a) Costs to capacities.



(b) Capacities to costs.

Fig. 5. Graphs transforming their edge weights to be interpreted as costs or capacities. The edges are labeled with their value, which also determines their thickness.

Secondly, and relatedly, it is always possible to transform costs into capacities, but the converse is not true. Compare Figure 5(a) with Figure 5(b). An edge with cost infinity has capacity zero (equivalent to an absent edge), and an existing edge with cost zero has infinite capacity (Figure 5(a)). However, an edge with infinite capacity cannot have a cost of zero – as we mention before, it would be confused with an absent edge in the adjacency matrix (Figure 5(b)). As a consequence, MLGE can be applied in all scenarios – just transform edge costs into capacities –, but MLGFT and MLEMD need some non-trivial adaptations to be applied to the edge-weight-as-capacity scenario.

None of this is insurmountable. One could always remember to keep track of the two semantically different zeroes in the edge weights in MLGFT and MLEMD and apply our node collapse strategy, but it represents a small argument in favor of using the cleaner interpretation of MLGE over the alternatives.

5 EXPERIMENTS

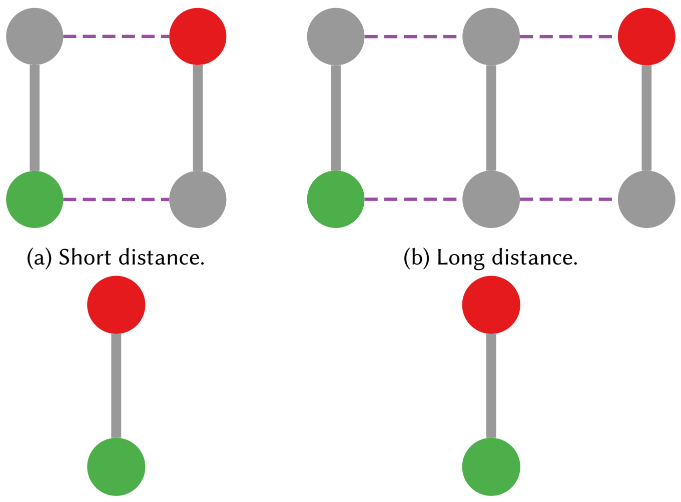
5.1 Setup

The experiments in this section use the following datasets:

- **EU Airlines** [4]. In this network, the nodes are airports, connected if an airline has a direct flight between the two airports. Each airline is a layer in the network. We clean the data by removing airports that are not actually in Europe, plus some generic airport codes like "XXXX". Each node has an attribute telling us in which country the airport is located.
- **Ego SM** is a network extracted in 2013 from various social media – each of which is a layer of the network. It collects all the direct friends – and the relations between them – of the author on Facebook, Twitter, Linkedin, Last.Fm, Flickr, Google+, and Gmail.
- **Copenhagen** is the data produced by a study on mobility and social networks in Denmark [44]. Students connect if they are Facebook friends, and if they call or text each other. Each of these relationships is a layer of the network. The original data for calls and texts is weighted and directed, but here we ignore the edge's weight and direction for simplicity.
- **Aarhus** is a network connecting 61 faculty members in the Aarhus university CS department in various ways, each determining a layer of the network [31]. They could collaborate, co-author, have lunch together, and more.
- **Physics** is a network of scientific co-authors [11]. Physicists connect if they co-author a paper, and the paper's classification determines the layer in which the connection appears. There are 16 total classifications.
- **IRA** is a network of social relationships between members of the Irish Republic Army terrorist group [17]. Relationships come from five different time periods between 1970 and 1998, each providing a layer in the network.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert D\rvert$	$\lvert C\rvert$	Dens
EU Airlines	2,034	3,588	37	11,611	0.001
Ego SM	1,031	6,119	7	780	0.012
Copenhagen	1,904	7,747	3	1,539	0.004
Aarhus	224	620	5	328	0.025
Physics	909	7,090	16	849	0.017
IRA	1,293	2,131	5	931	0.003

Table 1. The number of nodes ($|V|$), edges ($|E|$), layers ($|D|$), inter-layer couplings ($|C|$), and edge density (Dens) of all the cleaned datasets used in this paper.



(a) Short distance. (b) Long distance.

(c) Flattening of (a). (d) Flattening of (b).

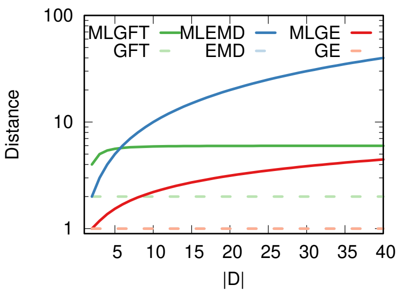
Fig. 6. The setup for our chain test, increasing the number of layers between source (red) and destination (green) node. Dashed purple edges represent the inter-layer couplings.

Table 1 reports their summary statistics. Here, the number of nodes $|V|$ is the actual number of nodes in the multilayer structure. Thus, two coupled nodes from different layers count as two distinct nodes, even though in the literature they would normally be counted as a single node. The inter-layer coupling count $|C|$ assumes a clique-style inter-layer coupling. If a node has no edges in a layer, it does not appear in that layer and thus needs no coupling.

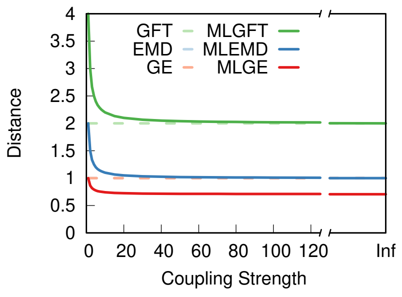
We perform all experiments on a Xeon E-2286M 2.40GHz CPU, 32GB of RAM, Ubuntu 20.04.2 LTS.

5.2 Chain Test Validation

In this section we validate our multilayer NVD measures. We do so by testing some intuitive scenarios, using chain of layers – which is why we call this the chain test. We start with a chain of layers. Each layer contains a single edge. The two nodes are coupled, via a chain coupling style, to the two nodes of the nearest layer. Figure 6(a) shows the setup with $|D| = 2$ layers and all coupling weights in C set to one. We then calculate the distance between two vectors occupying the node in each corner of the setup.

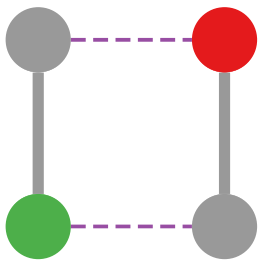


(a) By Increasing $|D|$.

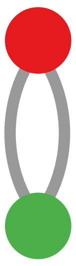


(b) By Increasing C Weights.

Fig. 7. The NVD distance value (y axis) for each of the alternative measures (line color), by varying the length (subfigure a, x axis) and the strength of the inter-layer couplings (subfigure b, x axis) in the multilayer chain test. Non-multilayer measures in dashed lines.



(a) Original network.



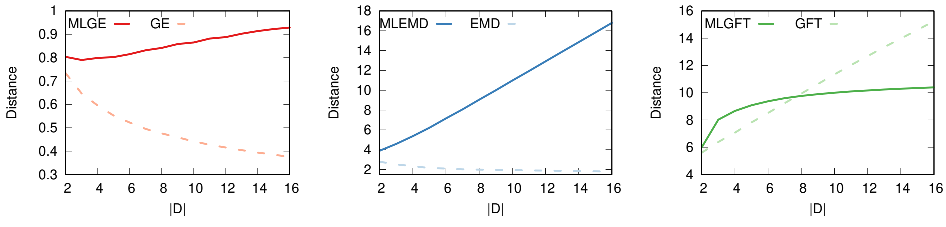
(b) Infinite strength C.

Fig. 8. The result of having infinite strength C couplings (in dashed purple edges).

5.2.1 By Increasing $|D|$. As we increase $|D|$ (Figure 6(b)), the distance between the two nodes increases, proportional to the cost of transitioning between layers. We thus expect an NVD measure to increase. However, if we did not have the capacity of seeing the multilayer structure, single-layer NVD measures would only see the nodes as always being directly connected (Figure 6(c-d)). We thus expect them to fail.

Figure 7(a) confirms our expectations. As we increase the number of layers between the origin and the destination, all multilayer measures (MLGE, MLEMD, and MLGFT) correctly report longer and longer distances. An analyst unable to see the multilayer nature of the data, instead, would estimate the distances as constant (using GE, EMD, or GFT). MLEMD is the most sensitive to the added layer: its distance has a linear relationship with $|D|$. MLGE has a sublinear relationship – specifically, it is proportional to $\sqrt{|D|}$, while MLGFT is the least sensitive.

5.2.2 By Increasing C Weights. When we increase the strength of the couplings C, keeping $|D| = 2$, MLGE behaves differently from MLEMD and MLGFT. If the strength of C is infinite, all coupled nodes collapse onto each other across layers, so we can approximate the multilayer network with a multigraph (Figure 8). Since MLGE simulates the process of reaching and equilibrium state between the input vectors, the fact that there are $|D|$ links between the two nodes is important. Thus, as the strength of C grows, MLGE converges to a distance estimation that is *lower* than the one you would get if the network would be single-layer, because alternative paths to achieve equilibrium exist (Figure 7(b)).



(a) MLGE vs GE. (b) MLEMD vs EMD. (c) MLGFT vs GFT.

Fig. 9. The Small World validation result: NVD distance (y axis) as we add more and more layers (x axis). Multilayer distance in dark color and single layer distances in dashed bright color.

This is not true for MLEMD and MLGFT. In this scenario, only the shortest path matters. The fact that there are two possible paths is irrelevant, because the amount of weight to transfer ($m_{a,b}$) and the distance to cover ($\delta_{a,b} = 1$) are the same, thus M is the same. This means that MLEMD and MLGFT initially think that the distance to cover in the multilayer network is larger than the one to cover in the single layer network, and only when the strength of coupling links is infinite we collapse exactly on the single layer case (note that, in this scenario GE = EMD, thus the EMD light blue data line is hidden behind GE's light red).

The behavior of MLGE here is more intuitive, because MLGE does not lose sight of the fact that, if two nodes connect in multiple layers, then they are more related to each other and thus should be able to pass information more easily.

5.3 Small World Validation

The chain test can show the behavior of the measure in a simple and intuitive setting, but it lacks in complexity. We complement it with a similar setup using a more complex model, namely the Watts-Strogatz Small World (SW) model [49]. In SW, nodes are displayed in a lattice and connected with a fixed number of their neighbors. Then, each connection has a small probability of being rewired, creating shortcuts in the lattice. We generate $|D|$ independent SW networks, each generating a layer of the network. Nodes with the same numeric id are coupled together across layers with a chain coupling.

In our test, v_1 has a value of 1 for four nodes at one end of the lattice in the first layer, while v_2 has a value of 1 for four nodes at the opposite end of the lattice in the last layer. If $|D| = 2$ the first and the last layers are adjacent, but if $|D| > 2$ then there are $|D| - 2$ intermediate layers between them. Thus, we expect a proper multilayer NVD measure to record larger and larger distances as $|D|$ grows.

On the other hand, a non multilayer NVD measure should shrink. This is because the layers are generated independently: when flattened, all created shortcuts will allow each node in the lattice to reach more and more portions of the lattice. Thus the two sets of nodes with 1 values in v_1 and v_2 will get closer and closer.

Figure 9 shows that our intuition is correct for the MLGE and MLEMD measures: the multilayer distance grows as we add more and more layers between v_1 and v_2, but the single layer measure – blind to this increase in complexity in the structure – decreases. The only single layer measure growing as the number of layers grows is GFT, which has the counterintuitive behavior of considering nodes farther away as the network gets denser.

5.4 Performance

Here we use NVD techniques to estimate the propagation of an event in a network, using three models: the cascade and the threshold models, both developed by Granovetter [19], and the Susceptible-Infected (SI) compartmentalized epidemic model on networks [37]. We use three models to show that some NVD measures are more suited to work in different scenarios.

The models have some common characteristics that make them a natural test for NVD measures. In all models we can represent the nodes' status as a vector containing 0 if the node is not active, and 1 if it is. No other values are possible. All models depend on some β parameter. How β is used is different in different models – which is why we will use different symbols to refer to the β of a specific model: β_c, β_t, and β_s for the cascade, threshold, and SI models, respectively. However, the common factor is that they all use their specific β to determine, at each time step, which nodes change their values from 0 to 1 – and from 1 to 0, if allowed by the model.

We run $10,000$ simulations of each model on each of our datasets. For each run, we infect 10-20 nodes at random – generating a binary node vector v_1 – and we extract a β value from a random distribution. We run each simulation for five steps. The final node configuration is stored in another node vector v_2. We then calculate the NVD between v_1 and v_2: the distance between the initial and the final statuses of the nodes.

We then estimate the contribution an NVD measure does to predicting the real β value. We do so in two steps. First, we run a regression between β and $|v_2|$, i.e. the number of infected nodes at the end of the process. This regression will have a certain mean absolute error $\hat{\beta}_{base}$. Then we run a regression predicting β with both $|v_2|$ and our NVD measure. This regression will have a – hopefully – lower mean absolute error $\hat{\beta}_{nvd}$. The Relative Mean Absolute Error Difference between the two models is:

$$RMAED = \frac{\hat{\beta}_{base} - \hat{\beta}_{nvd}}{\hat{\beta}_{base}}.$$

This formula tells us how much we were able to reduce, on average, the mean absolute error of the prediction of β by adding the NVD measure to our regression. Thus, the higher RMAED, the more useful NVD is. The RMAED value can be interpreted as a relative gain. If it is equal to 0.1, it means that, on average, we reduced the prediction error by 10%.

We need these two steps because $|v_2|$ is a strong predictor of β – if the propagation event can infect nodes more or less easily, this will be reflected in how many nodes are infected at the end. $|v_2|$ is a trivial quantity to calculate: to be useful, the more complex NVD measure must reduce the errors made by using only $|v_2|$.

Finally, we make two batches of comparisons. First, we compare MLGE against alternative multilayer network distance measures – the multilayer adaptations of EMD (MLEMD) and of GFT (MLGFT). Second, we flatten each G into its single layer version and we apply the single layer versions of the measures. This is done to prove that the multilayer structure is adding value to the analysis, recalling our argument from Section 3.2.

5.4.1 Cascade Model. In the cascade model we have a belief propagating on a network, whose spread is regulated by a parameter β_c. To be activated, a node needs at least a β_c fraction of its neighbors to be active as well. A node moves from inactive to active when this condition is met, and can move from active to inactive when this condition stops being met. There is an expected correlation between β_c and the network distance the belief will cover [10].

We extract β_c from a normal distribution. We fix the average and standard deviation of the distribution such that we avoid oversampling the β_c values that would cause the threshold model to

Network	MLGE	MLEMD	MLGFT	GE	EMD	GFT
EU Airlines	**0.358**	0.144	0.246	0.181	0.098	0.054
Ego SM	**0.283**	0.167	0.151	0.223	0.161	0.082
Copenhagen	**0.198**	0.015	0.020	0.031	-0.010	0.007
Aarhus	**0.017**	0.015	0.007	0.010	0.005	0.005
Physics	**0.221**	0.155	0.063	0.193	0.141	0.153
IRA	**0.187**	0.138	0.073	0.185	0.127	0.054

Table 2. The RMAED of each multilayer network distance measure (left panel) and single layer distance measure (right panel). Best performing method in bold. Cascade model.

Network	MLGE	MLEMD	MLGFT	GE	EMD	GFT
EU Airlines	0.219	**0.223**	0.200	0.091	0.103	0.033
Ego SM	**0.110**	0.109	0.094	0.041	0.043	0.028
Copenhagen	0.001	0.000	0.001	0.000	0.000	0.000
Aarhus	**0.120**	0.114	0.104	0.011	0.007	0.011
Physics	**0.030**	0.026	0.020	-0.002	-0.002	-0.002
IRA	0.167	**0.188**	0.170	0.135	0.133	0.093

Table 3. The RMAED of each multilayer network distance measure (left panel) and single layer distance measure (right panel). Best performing method in bold. Threshold model.

die out. High β_c values result in a final state with no active nodes, which are indistinguishable from each other. Since β_c is continuous and normally distributed, the quality measure RMAED is based on a simple linear regression model. In this experiment, we take the logarithm of the NVD measure. This is because, in the cascade model, each linear increase in β_c causes an exponential decrease in active nodes, because the requirements for being active for each node becomes demanding. Thus we would expect a log-linear relationship between β_c and the NVD distance which makes our validation via linear fit unsuitable. By log-transofming the NVD value, we are correctly estimating its mean absolute error.

Table 2 shows the results. Firstly, when comparing MLGE with MLEMD and MLGFT, we can see that MLGE shows a stronger predictive error loss than the alternatives. Secondly, for all measures, the loss in prediction power when working on a flattened structure is noticeable in all but one cases (MLGFT on the Physics network). This proves the usefulness of the multilayer extension to NVD.

5.4.2 Threshold Model. The threshold model works exactly like the cascade model, but β_t this time is not the relative number of neighbors that needs to be infected, but the absolute number. To be activated, a node needs at least β_t neighbors to be active. The higher the β_t the more active neighbors you need to transitions and, therefore, the less the covered distance.

We extract β_t from a discrete uniform random distribution. The minimum is 1, and the maximum (empirically, between 4 and 8) is a value that generates at least one additional infected node in the network. Since β_t is discrete and not normally distributed, we cannot run a linear model to predict it. Instead, we run a Poisson regression, which is used to model count variables. In this case, we do not log-transform the NVD measure, because this is already taken into account in the assumptions of the Poisson regression.

Table 3 shows the results. The table highlights that, depending on the underlying propagation model, different NVD measures might perform better or worse. Compared to Table 2, now we see that the difference between MLGE and MLEMD is much lower. In some cases, MLEMD is preferred.

Network	MLGE	MLEMD	MLGFT	GE	EMD	GFT
EU Airlines	0.029	**0.044**	0.006	0.004	0.004	0.000
Ego SM	**0.033**	0.030	0.007	0.012	0.006	0.003
Copenhagen	0.056	**0.086**	0.002	0.007	0.007	0.001
Aarhus	**0.013**	0.009	0.002	0.001	0.004	0.002
Physics	**0.015**	0.013	0.006	0.007	0.006	0.002
IRA	**0.039**	0.037	0.002	0.029	0.024	0.000

Table 4. The RMAED of each multilayer network distance measure (left panel) and single layer distance measure (right panel). Best performing method in bold. SI model.

We also see that the underlying network makes a difference. For the Copenhagen dataset, no NVD measure actually provides any improvement over simply counting the number of final active nodes. It is still true, however, that multilayer NVDs always perform better than ignoring the multilayer nature of the propagation phenomenon.

5.4.3 SI Model. Finally, we test an SI model, where the presence of a single active neighbor turns a node active with probability β_s. We extract β_s from a random normal distribution, and thus we perform a simple linear regression.

This is the hardest test for these measures, because there is a strong direct association between β_s and $|v_2|$. This is due to the fact that, in an SI model, nodes cannot turn inactive after they have been activated. To give context, the R^2 of the regression between β_s and $|v_2|$ is 0.917 in the EU Airlines network, i.e. $|v_2|$ is practically a perfect predictor of β_s.

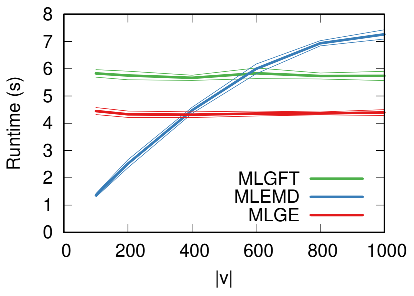
However, Table 4 shows that multilayer NVD measures can still reduce the mean absolute error by several percentage points, even in present of almost perfect correlations. Again, our proposed MLGE is the best contributor in most datasets, and multilayer NVD measures perform better than NVD measures that only access to single layer information.

In summary, in this section we see two things. First: multilayer NVD measures are always an improvement over monolayer NVD measures. Second: while MLGE is always a good multilayer NVD measure, it is not always the best. This is model specific: depending on the phenomenon under study one might want to choose MLEMD instead. This is a known property of NVD measures: they have different ways of modeling propagation on a network, as classified in previous work [10]. MLGE assumes a random percolation model, while MLEMD assumes an omniscient optimizer. Thus, the choice of the specific NVD measure is not only a matter of performance: it also depends on the characteristics of the precise phenomenon under study.

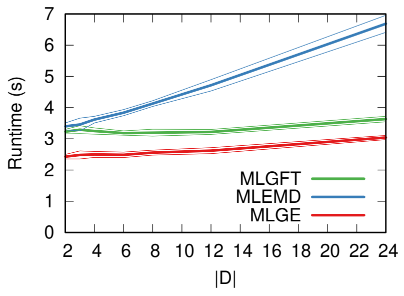
5.5 Efficiency

MLGE's efficiency is linked to the calculation of the pseudoinverse of the Laplacian, which is $O(|V|^{\omega})$, with ω higher than 2 but lower than 3, depending on the implementation of SVD used. Empirically, on the machine used for the experiments, a full run of MLGE on a network with $|V| \sim 1000$ and $|E| \sim 9000$ took 4.4 seconds. Figure 10 shows how that compares to the other methods on a number of dimensions. In all cases, we perform the experiments on a stochastic blockmodel synthetic network, which allows us to vary one aspect of the network while keeping the rest constant.

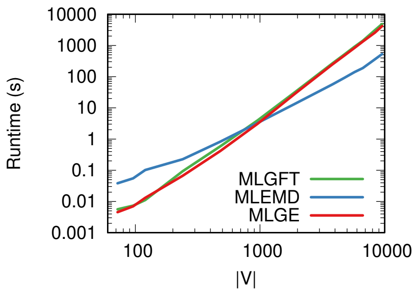
5.5.1 By Input Size. In Figure 10(a) we change the size of the input vectors v_x in terms of number of nonzero entries. We can see that both MLGE and MLGFT are unaffected by how many nonzero entries v_x have. MLGE has a slightly better constant factor in the runtime than MLGFT, as



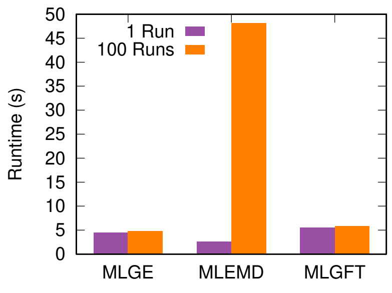
(a) By vector size.



(b) By number of layers.



(c) By number of nodes.



(d) Multiple runs.

Fig. 10. The runtimes (y axis) of all methods, varying different dimensions of the input (a-c). The thick line is the average of ten runs and the thin lines show the standard deviation. (d) Comparing one run against 100 runs on the same network, reusing computation.

calculating the pseudoinverse of the Laplacian is faster than calculating the eigenvectors. MLEMD, instead, is heavily affected, because it needs to calculate the shortest paths between all pairs of nodes with at least one non-zero entry in v_x. If there are many zeroes, it means that MLEMD has much fewer computations to make.

5.5.2 *By Layer Count.* In Figure 10(b) we keep the number of nodes and intra-layer edges constant, but we split the network in more layers. MLGE and MLGFT are unaffected, while MLEMD takes longer times. While the number of intra-layer edges is constant, there are more inter-layer couplings, and thus it takes longer to calculate the shortest paths.

5.5.3 *By Node Count.* In Figure 10(c) we increase the number of nodes in the network. We can see that, as the networks get larger, there is little difference in the order of magnitude of the runtime between the methods (note the logarithmic scaling). MLEMD scales better, because the shortest path computation is more efficient than pseudoinverting ($O(|V|^2)$ vs $O(|V|^{\sim 3})$), but EMD still needs to optimize the weight transfer, which is a computationally complex problem.

5.5.4 *Runtime Breakdown.* Figure 10(d) highlights the key advantage of MLGE and MLGFT over MLEMD. Here, we are in the scenario of having a single network on which a process unfolds, and many vectors v_x for which we are interested in calculating the distances. This is a common scenario: simulating day after day of an epidemic outbreak on the same network, the distance for every exporter pair on the unchanging Product Space, etc.

In both MLGE and MLGFT, most of the running time is taken by the expensive step of pseudoinverting or calculating the eigenvectors of the Laplacian. However, if the graph does not change,

Rank	Airline	Δ_d	$\Delta\vert E\vert$	$\Delta\vert V\vert$
1	Flybe	0.0557	88	6
2	Austrian Airlines	0.0180	43	2
3	Netjets	0.0177	143	14
4	Ryanair	0.0152	523	19
5	Turkish Airlines	0.0149	95	10

Table 5. Top five airlines by Δ_l value. We also report the number of edges ($\Delta|E|$) and of nodes ($\Delta|V|$) that G loses once we remove layer l.

both can be cached and reused. Then, calculating more distances adds little to no additional run-time. Calculating a single distance in MLGE in our scenario takes 4.4 seconds, but calculating 100 distances takes only 4.8 in total.

On the other hand, in MLEMD we can only cache the shortest paths, but we still need to find the optimal weight transfer between two vectors. As a result, even if calculating a single distance takes only 2.6 seconds, calculating 100 distances takes 48 seconds, ten times as much as with MLGE.

5.6 Case Study

In this case study, we focus on the EU Airlines dataset. We use MLGE to answer the question: how important is an airline for the connectivity of Europe? And: which are the most important airlines for the connectivity of a country in Europe? To answer these questions, we represent each country c as a node vector v_c. The vector v_c contains 1 for the entries corresponding to the airports located inside c, and 0 otherwise. We then calculate all country-country distances by applying MLGE to all country pairs, obtaining the current average distance between countries in the EU:

$$EU_{avg} = \frac{\sum\limits_{c1,c2} MLGE(G, v_{c1}, v_{c2})}{|O|(|O|-1)},$$

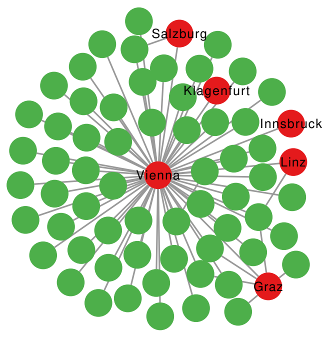
with G being the network, and O the set of countries. Note that we just skip calculating the distance of c with itself.

We can repeat the procedure removing each layer d in turn. This results in a EU_d distance estimation, which normally is higher – but it could be lower if removing d also drops a significant number of hard-to-reach nodes. The $\Delta_d = EU_d - EU_{avg}$ value is interesting: it tells us how much the distance increased by removing d from the network. The higher this value, the more important the layer is for the network.

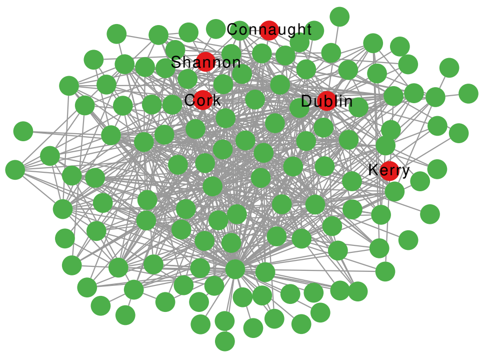
Table 5 reports the top five airlines according to this criterion. This ranking is interesting because we established that MLGE has a clean intuition as a distance – thus it is meaningful –, and it cannot be approximated with simpler statistics. One could expect that an airline is important proportionally to the number of edges it contributes. However, by far the layer with most edges is Ryanair (523 edges, the second largest layer is Easyjet with 199). Yet, Ryanair only ranks fourth in importance (Easyjet ranks seventh), and the top two layers have fewer than 100 edges.

In fact, the p-value of the expected anti-correlation between Δ_d and the size in edges of d is a mere 0.07 after controlling for $\Delta|V|$ – which we need to do because otherwise dropping edges could result in decreasing Δ_d if we also drop enough hard-to-reach nodes. This is far from the required significance threshold of 0.01.

One cannot also explain the Δ_d value by looking at the topology of the layer itself. Airlines tend to have two main wiring strategies: (i) connect everything to a central hub – usually a national airline strategy like for Austrian Airlines (Figure 11(a)); or (ii) have an "all-to-all" decentralized



(a) Austrian Airlines.



(b) Ryanair.

Fig. 11. Two layers of the EU Airlines network. Nodes in red are the domestic airports of the two companies.

Country	Airline	EU Δ_d Rank	Country Δ_d Rank
Belgium	Brussels Airlines	28th	2nd
Czechia	Czech Airlines	10th	2nd
France	Air France	17th	15th
Germany	Lufthansa	6th	3rd
Ireland	Aer Lingus	30th	3rd
Italy	Alitalia	19th	17th
Norway	Norwegian Air	18th	2nd
UK	British Airways	11th	10th

Table 6. Selected countries with the ranking of their domestic airline in the overall EU Δ_l ranks and in the ranking considering only the country as an origin.

strategy – a low-cost approach such as the one of Ryanair (Figure 11(b)). The difference between Figures 11(a) and 11(b) can be estimated by using a centralization score [16] – by comparing the centrality scores with the ones one would obtain from a maximally centralized graph, i.e. a star.

The p-values of the correlations between Δ_d and degree, closeness, and betweenness centralization scores are 0.26, 0.18, and 0.0336, respectively. When we perform the Holm-Bonferroni correction for multiple hypothesis tests [23], such p-values fail to clear the significance threshold of 0.1, meaning that there is no significant correlation between Δ_d and them. One cannot simply approximate Δ_d by looking at the centralization of the layer.

For our second question, we perform the same operation but, rather than making an average across all country-country pairs, we focus on each country and we average only its distances to all other EU countries. We expect to see that the national airline of the country ranking higher than it does for the whole Europe, as it connects all of its cities to the rest of the system.

Table 6 reports the result of this test for selected countries. In all cases our expectation is met: the flagship airline is more important for the country itself than for Europe as a whole, because it ranks higher in importance. However, there is a clear distinction between countries. For some (Belgium, Czechia, Ireland, Norway) the rank difference is large: between 8 and 27 positions. For others (France, Germany, Italy, UK), the rank difference is barely noticeable. This could be explained by the size and economic importance of the countries in the latter group, which causes non-domestic airlines to still serve them with a wide variety of routes.

6 CONCLUSION

In this paper we extend an existing algorithm to solve the Node Vector Distance (NVD) problem to handle multilayer networks. This is a necessary extension, because multilayer networks can accurately model more complex real world phenomena. Ignoring this more complex structure would lead to incorrect estimations. Our method, Multi Layer Generalized Euclidean (MLGE) can handle weighted networks with an arbitrary number of layers and an arbitrary function determining the strength of inter-layer couplings. The method has both theoretical and practical advantages over the possible alternatives. Specifically, by using an "edge weights as capacities" approach, it is more intuitive, and it is overall more effective in retrieving infection parameters in a threshold model. We also show that MLGE's results are intuitive for a human interpreter, and they are useful in a case study.

This is an improvement over the literature solving NVD, but a number of questions remain open. First, all methods – both single and multilayer – require the input network to be connected in a single component. How to generalize this method to arbitrary networks is still unclear. Second, using SVD to pseudoinvert the Laplacian is by no means the only possible choice. The growing literature on node embeddings [2, 35] could provide useful alternatives, which might also help with the high space and time complexity of the method. Further, one could use these methods to infer the underlying network structure behind a propagation event. Finally, one could extend the NVD literature by finding correlations between spreading events, rather than their distance – similarly to the network-event correlation explored in related works [20].

Acknowledgments. The author wishes to thank Clara Vandeweerdt for insightful discussions regarding the statistical analysis.

REFERENCES

[1] Anton Andreychuk, Konstantin Yakovlev, Dor Atzmon, and Roni Sternr. 2019. Multi-Agent Pathfinding with Continuous Time. In *IJCAI*, Vol. 19.
[2] Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* 15, 6 (2003), 1373–1396.
[3] Stefano Boccaletti, Ginestra Bianconi, Regino Criado, Charo I Del Genio, Jesús Gómez-Gardenes, Miguel Romance, Irene Sendina-Nadal, Zhen Wang, and Massimiliano Zanin. 2014. The structure and dynamics of multilayer networks. *Physics reports* 544, 1 (2014), 1–122.
[4] Alessio Cardillo, Jesús Gómez-Gardenes, Massimiliano Zanin, Miguel Romance, David Papo, Francisco Del Pozo, and Stefano Boccaletti. 2013. Emergence of network features from multiplexity. *Scientific reports* 3, 1 (2013), 1–6.
[5] Ronald R Coifman and Stéphane Lafon. 2006. Diffusion maps. *Applied and computational harmonic analysis* 21, 1 (2006), 5–30.
[6] Vittoria Colizza, Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. 2006. The role of the airline transportation network in the prediction and predictability of global epidemics. *PNAS* 103, 7 (2006), 2015–2020.
[7] Michele Coscia. 2020. Generalized Euclidean Measure to Estimate Network Distances. In *ICWSM*, Vol. 14. 119–129.
[8] Michele Coscia. 2021. The Atlas for the Aspiring Network Scientist. *arXiv e-prints* (2021), arXiv–2101.
[9] Michele Coscia. 2021. Pearson correlations on complex networks. *Journal of Complex Networks* 9, 6 (2021), cnab036.
[10] Michele Coscia, Andres Gomez-Lievano, James Mcnerney, and Frank Neffke. 2020. The Node Vector Distance Problem in Complex Networks. *ACM Computing Surveys (CSUR)* 53, 6 (2020), 1–27.
[11] Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. 2015. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Physical Review X* 5, 1 (2015), 011027.
[12] Karel Devriendt, Samuel Martin-Gutierrez, and Renaud Lambiotte. 2020. Variance and covariance of distributions on graphs. *arXiv preprint arXiv:2008.09155* (2020).
[13] Wendy Ellens, FM Spieksma, P Van Mieghem, A Jamakovic, and RE Kooij. 2011. Effective graph resistance. *Linear algebra and its applications* 435, 10 (2011), 2491–2506.
[14] Matthias Erbar, Martin Rumpf, Bernhard Schmitzer, and Stefan Simon. 2017. Computation of Optimal Transport on Discrete Metric Measure Spaces. *arXiv preprint arXiv:1707.06859* (2017).

[15] Montacer Essid and Justin Solomon. 2017. Quadratically-Regularized Optimal Transport on Graphs. *arXiv preprint arXiv:1704.08200* (2017).

[16] Linton C Freeman. 1978. Centrality in social networks conceptual clarification. *Social networks* 1, 3 (1978), 215–239.

[17] Paul Gill, Jeongyoon Lee, Karl R Rethemeyer, John Horgan, and Victor Asal. 2014. Lethal connections: The determinants of network connections in the Provisional Irish Republican Army, 1970–1998. *International Interactions* 40, 1 (2014), 52–78.

[18] Carsten Gottschlich and Dominic Schuhmacher. 2014. The shortlist method for fast computation of the earth mover's distance and finding optimal solutions to transportation problems. *PloS one* 9, 10 (2014), e110214.

[19] Mark Granovetter. 1978. Threshold models of collective behavior. *American journal of sociology* 83, 6 (1978), 1420–1443.

[20] Ziyu Guan, Jian Wu, Qing Zhang, Ambuj Singh, and Xifeng Yan. 2011. Assessing and ranking structural correlations in graphs. In *ACM SIGMOD*. 937–948.

[21] Ricardo Hausmann, César A Hidalgo, and al. 2014. *The atlas of economic complexity: Mapping paths to prosperity.* Mit Press.

[22] César A Hidalgo, Bailey Klinger, A-L Barabási, and Ricardo Hausmann. 2007. The product space conditions the development of nations. *Science* 317, 5837 (2007), 482–487.

[23] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[24] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. 2014. Multilayer networks. *Journal of complex networks* 2, 3 (2014), 203–271.

[25] Douglas J Klein and Milan Randić. 1993. Resistance distance. *Journal of mathematical chemistry* 12, 1 (1993), 81–95.

[26] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. 2013. Deltacon: A principled massive-graph similarity function. In *SDM*. SIAM, 162–170.

[27] Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. 2007. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)* 1, 1 (2007), 5–es.

[28] Jiaoyang Li, Pavel Surynek, Ariel Felner, Hang Ma, TK Satish Kumar, and Sven Koenig. 2019. Multi-agent path finding for large agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7627–7634.

[29] WUCHEN Li, ERNEST K Ryu, STANLEY Osher, WOTAO Yin, and WILFRID Gangbo. 2017. A parallel method for earth mover's distance. *UCLA CAM* (2017), 17–12.

[30] Minghua Liu, Hang Ma, Jiaoyang Li, and Sven Koenig. 2019. Task and Path Planning for Multi-Agent Pickup and Delivery. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1152–1160.

[31] Matteo Magnani, Barbora Micenkova, and Luca Rossi. 2013. Combinatorial analysis of multiple networks. *arXiv preprint arXiv:1303.4986* (2013).

[32] Victor Gabriel Lopez Mejia, Frank L Lewis, Yan Wan, Edgar N Sanchez, and Lingling Fan. 2019. Solutions for Multiagent Pursuit-Evasion Games on Communication Graphs: Finite-Time Capture and Asymptotic Behaviors. *IEEE Trans. Automat. Control* (2019).

[33] Bernardo Monechi, Alvaro Ruiz-Serrano, Francesca Tria, and Vittorio Loreto. 2017. Waves of novelties in the expansion into the adjacent possible. *PloS one* 12, 6 (2017), e0179303.

[34] Frank Neffke, Matté Hartog, Ron Boschma, and Martin Henning. 2018. Agents of structural change: The role of firms and entrepreneurs in regional diversification. *Economic Geography* 94, 1 (2018), 23–48.

[35] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1105–1114.

[36] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. 2010. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications* 1, 1 (2010), 19–30.

[37] Romualdo Pastor-Satorras and Alessandro Vespignani. 2001. Epidemic spreading in scale-free networks. *Physical review letters* 86, 14 (2001), 3200.

[38] Ofir Pele and Michael Werman. 2008. A linear time histogram metric for improved sift matching. In *ECCV*. Springer, 495–508.

[39] Ofir Pele and Michael Werman. 2009. Fast and robust earth mover's distances. In *ICCV*. IEEE, 460–467.

[40] Shmuel Peleg, Michael Werman, and Hillel Rom. 1989. A unified approach to the change of resolution: Space and gray-level. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11, 7 (1989), 739–742.

[41] Mason A Porter. 2018. What is… A multilayer network. *Notices of the AMS* 65, 11 (2018).

[42] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* 40, 2 (2000), 99–121.

[43] Aliaksei Sandryhaila and Jose MF Moura. 2014. Discrete Signal Processing on Graphs: Frequency Analysis. *IEEE Trans. Signal Processing* 62, 12 (2014), 3042–3054.

[44] Piotr Sapiezynski, Arkadiusz Stopczynski, David Dreyer Lassen, and Sune Lehmann. 2019. Interaction data from the copenhagen networks study. *Scientific Data* 6, 1 (2019), 1–10.

[45] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. 2012. A windowed graph Fourier transform. In *2012 IEEE Statistical Signal Processing Workshop (SSP)*. Ieee, 133–136.

[46] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. 2016. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis* 40, 2 (2016), 260–291.

[47] Nicholas M Stiffler and Jason M O'Kane. 2016. Pursuit-evasion with fixed beams. In *ICRA*. IEEE, 4251–4258.

[48] Yu-Xiang Wang, James Sharpnack, Alex Smola, and Ryan J Tibshirani. 2016. Trend filtering on graphs. *JMLR* 17, 105 (2016), 1–41.

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