Towards Optimal Connectivity on Multi-layered Networks

Chen Chen, Jingrui He, Nadya Bliss, and Hanghang Tong

Abstract—Networks are prevalent in many high impact domains. Moreover, cross-domain interactions are frequently observed in many applications, which naturally form the dependencies between different networks. Such kind of highly coupled network systems are referred to as *multi-layered networks*, and have been used to characterize various complex systems, including critical infrastructure networks, cyber-physical systems, collaboration platforms, biological systems and many more. Different from single-layered networks where the functionality of their nodes is mainly affected by within-layer connections, multi-layered networks are more vulnerable to disturbance as the impact can be amplified through cross-layer dependencies, leading to the cascade failure to the entire system. To manipulate the connectivity in multi-layered networks, some recent methods have been proposed based on *two-layered networks* with *specific* types of connectivity measures.
In this paper, we address the above challenges in multiple dimensions. First, we propose a family of connectivity measures (SubLine) that unifies a wide range of classic network connectivity measures. Third, we reveal that the connectivity measures in SubLine family enjoy *diminishing returns property*, which guarantees a near-optimal solution with linear complexity for the connectivity optimization problem. Finally, we evaluate our proposed algorithm on real data sets to demonstrate its effectiveness and efficiency.

Index Terms—Network Connectivity, Multi-layered Networks.

1 Introduction

Networks naturally arise from many high impact domains. Moreover, the cross-domain interactions between networks are frequently observed in many applications. The resulting inter-dependent networks naturally form a type of *multi-layered networks* [1], [2], [3], [4]. Critical infrastructure system is a classic example for multi-layered network as shown in Fig. 1. In this system, the power stations in the power grid are used to provide electricity to routers in the autonomous system network (AS network) and vehicles in the transportation network; while the AS network in turn is needed to provide communication mechanisms to keep power grid and transportation network work in order. On the other hand, for some coal-fired or gas-fired power stations, a well-functioning transportation network is required to supply fuel for those power stations. Therefore, the inter-dependent three layers in the system form a triangular dependency network. Another example is the organization-level collaboration platform, where the *team network* is supported by the *social network*, connecting its employee pool, which further interacts with the *information network*, linking to its knowledge base. Furthermore, the *social network* layer could have an embedded multi-layered structure (e.g., each of its layers represents a different collaboration type among different individuals); and so does the *information network*. In this application, the different layers form a tree-structured dependency network rooted on the team network layer.

Different from single-layered networks, multi-layered networks are more vulnerable to external attacks because their nodes can be affected by both within-layer connections and cross-layer dependencies. That is, even a small disturbance in one layer/network may be amplified in all its dependent networks through cross-layer dependencies, and cause cascade failure to the entire system. For example, when the supporting facilities (e.g., power stations) in a metropolitan area are destroyed by natural disasters like hurricanes or earthquakes, the resulting blackout

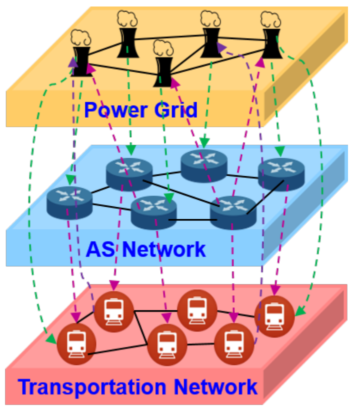


Fig. 1. A simplified example of multi-layered network.

would not only put tens of thousands of people in dark for a long time, but also paralyze the telecom network and cause a great interruption on the transportation network. Therefore, it is of key importance to identify crucial nodes in the supporting layer/network, whose loss would lead to a catastrophic failure of the entire system, so that the counter measures can be taken proactively. On the other hand, accessibility issues extensively exist in multi-layered network mining tasks. To manipulate the connectivity in layers with limited accessibility, one can only operate via the nodes from accessible layers that have large impact to target layers. Taking the multi-layered network depicted in Fig. 2(a) for example, assume that the only accessible layer in the system is the control layer and the goal is to minimize the connectivity in the satellite communication layer and physical layer simultaneously under k attacks, the only strategy we could adopt is to select a set of k nodes from the control layer, whose failure would cause largest reduction on the connectivity of the two target layers.

To tackle the connectivity optimization[1] problem in multi-layered networks, great efforts have been made from different research area for manipulating *two-layered* interdependent net-

• *C. Chen, J. He, N. Bliss and H. Tong are with Arizona State University, Tempe, AZ, 85281, USA.*
E-mail: {chen_chen, jingrui.he, nadya.bliss, hanghang.tong}@asu.edu

1. In this paper, connectivity optimization problem is defined as minimizing the connectivity of a target layer by removing a fixed number of nodes in the control layer (refer to the detailed definition in Section 4).

work systems [1], [2], [4], [5], [6]. Although much progress has been made, two key challenges have largely remained open. First (**connectivity measures**), there does not exist one single network connectivity measure that is superior to all other measures; but rather several connectivity measures are prevalent in the literature (e.g., robustness [7], vulnerability [8], triangle counts). Each of the existing optimization algorithms on multi-layered networks is tailored for one specific connectivity measure. It is not clear if an algorithm designed for one specific connectivity measure is still applicable to other measures. So how can we design a generic optimization strategy that applies to a variety of prevalent network connectivity measures? Second (**connectivity optimization**), an optimization strategy tailored for two-layered networks might be sub-optimal, or even misleading to multi-layered networks, e.g., in case we want to simultaneously optimize the connectivity in multiple layers by manipulating one common supporting layer. On the theoretic side, the optimality of the connectivity optimization problem of generic multi-layered networks is largely unknown.

This paper aims to address *all* these challenges, and the main contributions can be summarized as

- *Connectivity Measures*. We unify a family of prevalent network connectivity measures (SUBLINE), which are in close relation to a variety of important network parameters (e.g., epidemic threshold, network robustness, triangle capacity).
- *Connectivity Optimization*. We show that for *any* network connectivity measures in the SUBLINE family, the connectivity optimization problem with the proposed MULAN model enjoys the diminishing returns property, which naturally lends itself to a family of provable near-optimal optimization algorithms with linear complexity.
- *Empirical Evaluations*. We perform extensive experiments based on real data sets to validate the effectiveness and efficiency of the proposed algorithms.

The rest of the paper is organized as follows: Section 2 provides the background of multi-layered network model (MULAN). Section 3 gives the definition of a set of unified connectivity measures (SUBLINE) and some of its examples. In Section 4, we define the connectivity optimization problem in multi-layered network and propose its solutions. Section 5 evaluates the proposed algorithms. Section 6 briefly introduces related work on network connectivity and multi-layered network. Section 7 concludes the whole paper.

2 THE MULTI-LAYERED NETWORK MODEL

In this section, we introduce the multi-layered network model that admits an arbitrary number of layers with arbitrary dependency structure among different layers. We start with the main symbols used throughout the paper (Table 1). We use bold upper case letters for matrices (e.g., $\mathbf{A}$, $\mathbf{B}$), bold lower case letters for column vectors (e.g., $\mathbf{a}$, $\mathbf{b}$) and calligraphic font for sets (e.g., $\mathcal{A}$, $\mathcal{B}$). The transpose of a matrix is denoted with a prime, i.e., $\mathbf{A}'$ is the transpose of matrix $\mathbf{A}$.

With the above notation, we use the following definition of multi-layered networks as in [9].

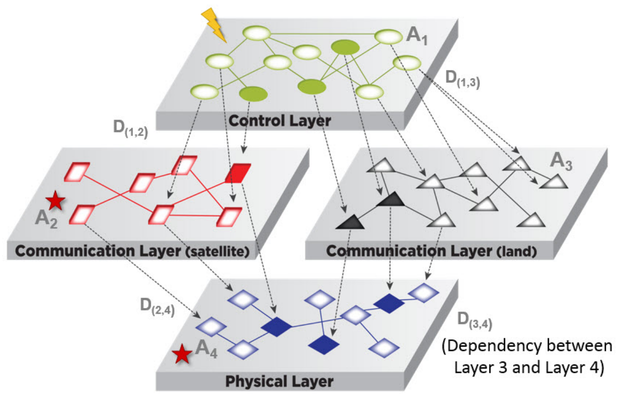
Definition 1. ***A Multi-layered Network Model (*MULAN*)****. Given (1) a binary $g \times g$ abstract layer-layer dependency network $\mathbf{G}$, where $\mathbf{G}(i,j) = 1$ indicates layer-j depends on layer-i (or layer-i supports layer-j), $\mathbf{G}(i,j) = 0$ means that there is no direct dependency from layer-i to layer-j; (2) a set of within-layer adjacency matrices $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_g\}$; (3) a set of cross-layer node-node dependency matrices $\mathcal{D}$, indexed by pair (i,j), $i,j \in [1,\ldots,g]$, such that for a pair (i,j), if $\mathbf{G}(i,j) = 1$, then $\mathbf{D}_{(i,j)}$ is a $n_i \times n_j$ matrix; otherwise $\mathbf{D}_{(i,j)} = \Phi$ (i.e., an empty set); (4) θ is a one-to-one mapping function that maps each node in layer-layer dependency network $\mathbf{G}$ to the corresponding within-layer adjacency matrix $\mathbf{A}_i$ $(i = 1,...,g)$; (5) φ is another one-to-one mapping function that maps each edge in $\mathbf{G}$ to the corresponding cross-layer node-node dependency matrix $\mathbf{D}_{(i,j)}$. We define a multi-layered network as a quintuple $\Gamma =< \mathbf{G}, \mathcal{A}, \mathcal{D}, \theta, \varphi >$.*

TABLE 1
Main Symbols.

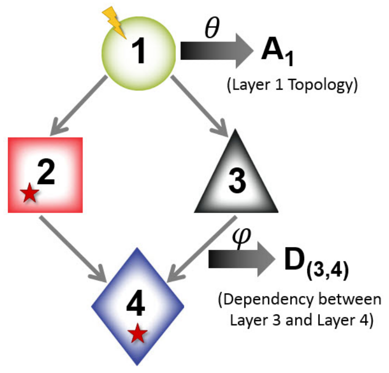
Symbol	Definition and Description
$\mathbf{A}, \mathbf{B}$	the adjacency matrices (bold upper case)
$\mathbf{a}, \mathbf{b}$	column vectors (bold lower case)
$\mathcal{A}, \mathcal{B}$	sets (calligraphic)
$\mathbf{A}(i,j)$	the element at i^{th} row j^{th} column in matrix $\mathbf{A}$
$\mathbf{A}(i,:)$	the i^{th} row of matrix $\mathbf{A}$
$\mathbf{A}(:,j)$	the j^{th} column of matrix $\mathbf{A}$
$\mathbf{A}'$	transpose of matrix $\mathbf{A}$
$\mathbf{G}$	the layer-layer dependency matrix
$\mathcal{A}$	networks at each layer of MULAN $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_g\}$
$\mathcal{D}$	cross-layer node-node dependency matrices
θ, φ	one to one mapping functions
Γ	multi-layered network MULAN $\Gamma =< \mathbf{G}, \mathcal{A}, \mathcal{D}, \theta, \varphi >$
$\mathcal{S}_i, \mathcal{T}_i, \ldots$	node sets in layer $\mathbf{A}_i$(calligraphic)
$\mathcal{S}_{i\to j}$	nodes in $\mathbf{A}_j$ that depend on nodes $\mathcal{S}$ in $\mathbf{A}_i$
$\mathcal{N}(\mathcal{S}_i)$	nodes and cross-layer links that depend on $\mathcal{S}_i$
m_i, n_i	number of edges and nodes in layer $\mathbf{A}_i$
$\lambda_{<\mathbf{A},j>}, \mathbf{u}_{<\mathbf{A},j>}$	j^{th} largest eigenvalue (in module) and its corresponding eigenvector of network $\mathbf{A}$
$\lambda_{\mathbf{A}}, \mathbf{u}_{\mathbf{A}}$	first eigenvalue and eigenvector of network $\mathbf{A}$
$C(\mathbf{A})$	connectivity function of network $\mathbf{A}$
$I_{\mathbf{A}}(\mathcal{S}_i)$	impact of node set $\mathcal{S}_i$ on network $\mathbf{A}$
$\mathbb{I}(\mathcal{S}_i)$	overall impact of node set $\mathcal{S}_i$ on MULAN

For simplicity, we restrict the within-layer adjacency matrices $\mathbf{A}_i$ to be simple (i.e., no self-loops), symmetric and binary; and the extension to the weighted, asymmetric case is straightforward. In this paper, we require cross-layer dependency network $\mathbf{G}$ to be an un-weighted graph with arbitrary dependency structure. Notice that compared with the existing pair-wise two-layered models, MULAN allows a much more flexible and complicated dependency structure among different layers. For the cross-layer node-node dependency matrix $\mathbf{D}_{(i,j)}$, $\mathbf{D}_{(i,j)}(s,t) = 1$ indicates that node s in layer i supports node t in layer j.

Fig. 2(a) presents an example of a four-layered network. In this example, layer-1 (e.g., the control layer) is the supporting layer (i.e., the root node in the layer-layer dependency network $\mathbf{G}$). Layer-2 and layer-3 directly depend on layer-1 (e.g., one represents a communication layer by satellites and the other represents another communication layer in landlines, respectively), while layer-4 (e.g., the physical layer) depends on both communication layers (layer-2 and layer-3). The abstracted layer-layer dependency network ($\mathbf{G}$) is shown in Fig. 2(b). $\mathcal{A} = \{\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4\}$ denotes the within-layer adjacency matrices, each of which describes the network topology in the corresponding layer. In this example, $\mathcal{D}$ is a set of matrices containing only four non-empty matrices: $\mathbf{D}_{(1,2)}, \mathbf{D}_{(1,3)}, \mathbf{D}_{(2,4)}$, and $\mathbf{D}_{(3,4)}$. For example, $\mathbf{D}_{(3,4)}$ describes the node-node dependency between layer-3 and layer-4. The one-to-one mapping function θ maps node 1 (i.e., Layer 1) in $\mathbf{G}$ to the within-layer adjacency matrix of layer-1 ($\mathbf{A}_1$); and the one-to-one mapping function φ maps edge $< 3,4 >$ in $\mathbf{G}$ to the cross-layer node-node dependency matrix $\mathbf{D}_{(3,4)}$ as shown in Fig. 2(b).


A1
D(1,3)
D(1,2)
Control Layer
A2
A3
Communication Layer (satellite)
Communication Layer (land)
D(2,4)
A4
D(3,4)
(Dependency between Layer 3 and Layer 4)
Physical Layer

(a) A four-layered network



(b) The corrsponding layer-layer dependency network $\mathbf{G}$

Fig. 2. An illustrative example of MULAN model

3 Unification of Connectivity Measures

In this section, we present a unified view for a variety of prevalent network connectivity measures.

3.1 Introducing SubLine Family

The key of our unified connectivity measure (referred to as SUBLINE in this paper) is to view the connectivity of the entire network as an aggregation over the connectivity measures of its sub-networks (e.g., subgraphs), that is

$$C(\mathbf{A}) = \sum_{\pi \subseteq \mathbf{A}} f(\pi) \tag{1}$$

where π is a subgraph of $\mathbf{A}$. The non-negative function $f : \pi \rightarrow \mathbb{R}^+$ maps any subgraph in $\mathbf{A}$ to a non-negative real number and $f(\phi) = 0$ for empty set ϕ. In other words, we view the connectivity of the entire network ($C(\mathbf{A})$) as the sum of the connectivity of all the subgraphs ($f(\pi)$). Based on such connectivity definition, we further define the impact function of a given set of nodes $\mathcal{S}$ as follows

$$I(\mathcal{S}) = C(\mathbf{A}) - C(\mathbf{A} \setminus \mathcal{S}) \tag{2}$$

where $\mathbf{A} \setminus \mathcal{S}$ is the residual network after removing node set $\mathcal{S}$ from the original network $\mathbf{A}$.

In multi-layered networks, as the functionality of each node depends on (1) the well-functioning of its depended node(s) and (2) its within-layer connections, the impact of node set $\mathcal{S}_i$ on the connectivity of layer-j can be quantified as the impact of all its dependents (either directly or indirectly) on the connectivity of layer-j (i.e. $I(\mathcal{S}_{i\rightarrow j})$). In the example in Fig. 2(a), the impact of $\mathcal{S}_1$ on layer-4 is $I(\mathcal{S}_{1\rightarrow 4}) = I((\mathcal{S}_{1\rightarrow 2})_{2\rightarrow 4} \cup (\mathcal{S}_{1\rightarrow 3})_{3\rightarrow 4})$. Based on eq. (2), we can define the overall impact of node set $\mathcal{S}_i$ in $\mathbf{A}_i$ on the multi-layered network system as

$$\mathbb{I}(\mathcal{S}_i) = \sum_{j=1}^{g} \alpha_j I(\mathcal{S}_{i\rightarrow j}) = \sum_{j=1}^{g} \alpha_j (C(\mathbf{A}_j) - C(\mathbf{A}_j \setminus \mathcal{S}_{i\rightarrow j})) \tag{3}$$

where $\alpha = [\alpha_1, ..., \alpha_g]'$ is a $g \times 1$ non-negative weight vector that assigns different weights to different layers in the system, which is a pre-defined parameter depending on the application task.

It is worth to mention that motifs (defined in [10]) are sub-networks as well. By setting function f as non-negative constants, many prevalent network connectivity measures can be reduced to SUBLINE connectivity measures; and we give three prominent examples below, including (1) the path capacity; (2) the loop capacity; and (3) the triangle capacity.

3.2 Example #1: Path Capacity

A natural way to measure network connectivity is through *path capacity*, which measures the total (weighted) number of paths in the network. In this case, the corresponding function $f()$ can be defined as follows.

$$f(\pi) = \begin{cases} \beta^{len(\pi)} & \text{if } \pi \text{ is a valid path of length } len(\pi) \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

where β is a damping factor between $(0, 1/\lambda_{\mathbf{A}})$ to penalize longer paths. With such a $f()$ function , the connectivity function $\mathbf{C}(\mathbf{A})$ defined in eq. (1) can be written as

$$\mathbf{C}(\mathbf{A}) = \mathbf{1}'(\sum_{t=1}^{\infty} \beta^t \mathbf{A}^t)\mathbf{1} = \mathbf{1}'(\mathbf{I} - \beta\mathbf{A})^{-1}\mathbf{1} \tag{5}$$

Remarks. We can also define the path capacity with respect to a given path length t as $\mathbf{C}(\mathbf{A}) = \mathbf{1}'\mathbf{A}^t\mathbf{1}$. When $t = 1$, $\mathbf{C}(\mathbf{A})$ is reduce to the edge capacity (density) of the graph, which is an important metric for network analysis. On the other hand, the 'average' path capacity $(\mathbf{1}'\mathbf{A}^t\mathbf{1})^{1/t}$ of a network converges to the leading eigenvalue of its adjacency matrix, i.e., $(\mathbf{1}'\mathbf{A}^t\mathbf{1})^{1/t} \xrightarrow{t\rightarrow\infty} \lambda_{\mathbf{A}}$, which is an important network vulnerability measure in relation to the so-called epidemic threshold [8]. In this case, the impact function of node set $\mathcal{S}$ in network $\mathbf{A}$ can be written as $I(\mathcal{S}) = \lambda_{\mathbf{A}} - \lambda_{\mathbf{A}\setminus\mathcal{S}}$, which can be further approximated by the so-called '*Shield-value*' score in [8] as follows

$$I_{\mathbf{A}}(\mathcal{S}) \approx \text{Sv}(\mathcal{S}) = \sum_{i\in\mathcal{S}} 2\lambda_{\mathbf{A}}\mathbf{u}_{\mathbf{A}}(i)^2 - \sum_{i,j\in\mathcal{S}} \mathbf{A}(i,j)\mathbf{u}_{\mathbf{A}}(i)\mathbf{u}_{\mathbf{A}}(j) \tag{6}$$

Thus, the overall impact of set $\mathcal{S}_i$ on MULAN can be estimated as

$$\mathbb{I}(\mathcal{S}_i) = \sum_{j=1}^{g} \alpha_j \text{Sv}(\mathcal{S}_{i\rightarrow j}) \tag{7}$$

3.3 Example #2: Loop Capacity

Another important way to measure network connectivity is through the loop capacity, which measures the total (weighted) number of loops in the network. In this case, the corresponding function $f()$ can be defined as follows.

$$f(\pi) = \begin{cases} 1/len(\pi)! & \text{if } \pi \text{ is a valid loop of length } len(\pi) \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Then, the connectivity function $\mathbf{C}(\mathbf{A})$ can be written as

$$\mathbf{C}(\mathbf{A}) = \sum_{t=1}^{\infty} \frac{1}{t!} trace(\mathbf{A}^t) = \sum_{i=1}^{n} e^{\lambda_{<\mathbf{A},i>}} \tag{9}$$

Accordingly, the impact function of a set of nodes $\mathcal{S}$ on network $\mathbf{A}$ is

$$I_{\mathbf{A}}(\mathcal{S}) = \sum_{i=1}^{n} e^{\lambda_{<\mathbf{A},i>}} - \sum_{i=1}^{n} e^{\lambda_{<\mathbf{A}\setminus\mathbf{S},i>}} \tag{10}$$

and the overall impact of set $\mathbf{S}_i$ on MuLaN can be calculated as

$$\mathbb{I}(\mathcal{S}) = \sum_{j=1}^{g} \alpha_j (\sum_{i=1}^{n_j} e^{\lambda_{<\mathbf{A}_j,i>}} - \sum_{i=1}^{n_j} e^{\lambda_{<\mathbf{A}_j\setminus\mathbf{S}_{i\to j},i>}}) \tag{11}$$

Remarks. The spectrum of a real network is often skewed. Thus, instead of computing all the eigenvalues of $\mathbf{A}$, a computationally much more efficient way is to compute the 'truncated' loop capacity by only keeping the top-r largest eigenvalues in the above equation, i.e., $\mathbf{C}(\mathbf{A}) = \sum_{i=1}^{r} e^{\lambda_{<\mathbf{A},i>}}$. Moreover, some recent work has suggested to adopt the logarithm of the truncated loop capacity as an alternative way to measure network robustness, with several distinctive advantages over the existing network robustness measures [7].

3.4 Example #3: Triangle Capacity

A localized network connectivity measure is through triangle capacity, i.e., the total number of triangles in the given network. In this case, the function $f()$ can be defined as

$$f(\pi) = \begin{cases} 1 & \text{if } \pi \text{ is a triangle} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

It has been shown in [11] that the number of triangles in a network is proportional to the sum of cubic of its eigenvalues. Thus, our corresponding connectivity function can be expressed as

$$\mathbf{C}(\mathbf{A}) = \sum_{i=1}^{n} \lambda^3_{<\mathbf{A},i>} \tag{13}$$

Then, the impact of node set $\mathcal{S}$ on the triangle capacity can be written as

$$I_{\mathbf{A}}(\mathcal{S}) = \sum_{i=1}^{n} \lambda^3_{<\mathbf{A},i>} - \sum_{i=1}^{n} \lambda^3_{<\mathbf{A}\setminus\mathcal{S},i>} \tag{14}$$

and the overall impact of set $\mathcal{S}_i$ on MuLaN is

$$\mathbb{I}(\mathcal{S}) = \sum_{j=1}^{g} \alpha_j (\sum_{i=1}^{n_j} \lambda^3_{<\mathbf{A}_j,i>} - \sum_{i=1}^{n_j} \lambda^3_{<\mathbf{A}_j\setminus\mathcal{S}_{i\to j},i>}) \tag{15}$$

Remarks. Similar to the loop capacity, we can use the truncated triangle capacity by only keeping the top-r eigenvalues to estimate the number of triangles.

4 Connectivity Optimization

In this section, we first define the connectivity optimization problem (Opera) on multi-layered network model (MuLaN); then unveil its major theoretic properties; and last propose a generic algorithmic framework to solve it.

4.1 Opera: Problem Statement

We formally define the connectivity optimization problem (Opera) on the proposed MuLaN model for multi-layered networks as follows.

Problem 1. Opera on MuLaN
Given: *(1) a multi-layered network* $\Gamma =< \mathbf{G}, \mathcal{A}, \mathcal{D}, \theta, \varphi >$*; (2) a control layer* $\mathbf{A}_l$*; (3) an impact function* $\mathbb{I}(.)$*; and (4) an integer* k *as operation budget;*
Output: *a set of* k *nodes* $\mathcal{S}_l$ *from the control layer* ($\mathbf{A}_l$) *such that* $\mathbb{I}(\mathcal{S}_l)$ *(the overall impact of* $\mathcal{S}_l$*) is maximized.*

In the above definition, the control layer $\mathbf{A}_l$ indicates the sources of the 'attack'; and the $g \times 1$ vector α indicates the target layer(s) as well as their relative weights. For instance, in Fig. 2(a), we can choose layer-1 as the control layer (indicated by the strike sign); and set $\alpha = [0\ 1\ 0\ 1\ ]'$, which means that both layer-2 and layer-4 are the target layers (indicated by the star signs) with equal weights between them. In this example, once a subset of nodes $\mathcal{S}$ in layer-1 are attacked/deleted (e.g., shaded circle nodes), all the nodes from layer-2 and layer-3 that are dependent on $\mathcal{S}$ (e.g., shaded parallelogram and triangle nodes) will be disabled/deleted, which will in turn cause the disfunction of the nodes in layer-4 (e.g., shaded diamond nodes) that depend on the affected nodes in layer-2 or layer-3. Our goal is to choose k nodes from layer-1 that have the maximal impact on both layer-2 and layer-4, i.e., to simultaneously decrease the connectivity $C(\mathbf{A}_2)$ and $C(\mathbf{A}_4)$ as much as possible.

4.2 Opera: Theory

In this subsection, we present the major theoretical results of the connectivity optimization problem (Opera) on multi-layered networks defined in Problem 1. It says that for **any** connectivity function $C(\mathbf{A})$ in the SubLine family (eq. (1)), for **any** multi-layered network in the MuLaN family (Definition 1), the connectivity optimization problem (Opera, Problem 1) bears **diminishing returns property**.

Let us start with the base case, where there is only one single input network. In this case, $\Gamma =< \mathbf{G}, \mathcal{A}, \mathcal{D}, \theta, \varphi >$ in Problem 1 degenerates to a single-layered network $\mathbf{A}$, which is both the control layer as well as the sole control target (i.e., $\alpha = 1$, and $l = 1$). With such a setting, Lemma 1 says that Opera enjoys the *diminishing returns property*, that is, the overall impact function $\mathbb{I}(\mathcal{S}_1)$ (which in this case degenerates to $I(\mathcal{S})$, i.e., the impact of the node set $\mathcal{S}$ on network $\mathbf{A}$ itself) is (a) monotonically non-decreasing; (b) sub-modular; and (c) normalized.

Lemma 1. Diminishing Returns Property of a Single-layered Network. *Given a simple undirected, un-weighted network* $\mathbf{A}$*, for any connectivity function* $C(\mathbf{A})$ *in the* SubLine *family, the impact function* $I(\mathcal{S})$ *is (a) monotonically non-decreasing; (b) sub-modular; and (c) normalized, where* $\mathcal{S} \subseteq \mathbf{A}$.

Proof. By the definition of the connectivity function $C(\mathbf{A})$ (eq. (1)), we have

$$I(\mathcal{S}) = \sum_{\pi\subseteq\mathbf{A}} f(\pi) - \sum_{\pi\subseteq\mathbf{A}\setminus\mathcal{S}} f(\pi) = \sum_{\pi\subseteq\mathbf{A},\ \pi\cap\mathcal{S}\neq\Phi} f(\pi)$$

where Φ is the empty set. Apparently, we have $I(\Phi) = 0$ since $f(\Phi) = 0$. In other words, the impact function $I(\mathcal{S})$ is normalized.

Let $\mathcal{I}, \mathcal{J}, \mathcal{K}$ be three sets and $\mathcal{I} \subseteq \mathcal{J}$. We further define three sets as follows: $\mathcal{S} = \mathcal{I} \cup \mathcal{K}, \quad \mathcal{T} = \mathcal{J} \cup \mathcal{K}, \quad \mathcal{R} = \mathcal{J} \setminus \mathcal{I}$.

We have

$$\begin{aligned} I(\mathcal{J}) - I(\mathcal{I}) &= \sum_{\pi\subseteq\mathbf{A},\ \pi\cap\mathcal{J}\neq\Phi} f(\pi) - \sum_{\pi\subseteq\mathbf{A},\ \pi\cap\mathcal{I}\neq\Phi} f(\pi) \\ &= \sum_{\pi\subseteq\mathbf{A},\ \pi\cap(\mathcal{J}\setminus\mathcal{I})\neq\Phi} f(\pi) = \sum_{\pi\subseteq\mathbf{A},\ \pi\cap\mathcal{R}\neq\Phi} f(\pi) \\ &\geq 0 \end{aligned}$$

which proves the monotonicity of the impact function $I(\mathcal{S})$.

Let us define another set $\mathcal{P} = \mathcal{T} \setminus \mathcal{S}$. We have that $\mathcal{P} = (\mathcal{J} \cup \mathcal{K}) \setminus (\mathcal{I} \cup \mathcal{K}) = \mathcal{R} \setminus (\mathcal{R} \cap \mathcal{K}) \subseteq \mathcal{R} = \mathcal{J} \setminus \mathcal{I}$. Then, we have

$$I(\mathcal{T}) - I(\mathcal{S}) = \sum_{\pi\subseteq\mathbf{A},\ \pi\cap\mathcal{P}\neq\Phi} f(\pi) \leq I(\mathcal{J}) - I(\mathcal{I})$$

which completes the proof of the sub-modularity of the impact function $I(\mathcal{S})$. □

In order to generalize Lemma 1 to an arbitrary, generic member in the MULAN family, we first need the following lemma, which says that the set-ordering relationship in a supporting layer is preserved through dependency links in all dependent layers of MULAN.

Lemma 2. Set-ordering Preservation Lemma on DAG. *Given a multi-layered network $\Gamma =< \mathbf{G}, \mathcal{A}, \mathcal{D}, \theta, \varphi >$ with the within-layer adjacency matrices $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_g\}$, and the dependency network $\mathbf{G}$ is a directed acyclic graph (DAG). For two node sets $\mathcal{I}_l, \mathcal{J}_l$ in $\mathbf{A}_l$ such that $\mathcal{I}_l \subseteq \mathcal{J}_l$, we have that in any layer $\mathbf{A}_i$ in the system, $\mathcal{I}_{l\to i} \subseteq \mathcal{J}_{l\to i}$ holds, where $\mathcal{I}_{l\to i}$ and $\mathcal{J}_{l\to i}$ are the node sets in layer $\mathbf{A}_i$ that depend on $\mathcal{I}_l$ and $\mathcal{J}_l$ in layer $\mathbf{A}_l$ respectively.*

Proof. If $l = i$, we have $\mathcal{J}_{l\to i} = \mathcal{J} \subseteq \mathcal{I}_{l\to i} = \mathcal{I}$ and Lemma 2 holds.

Second, if layer-*i* does not depend on layer-*l* either directly or indirectly, we have $\mathcal{J}_{l\to i} = \mathcal{I}_{l\to i} = \Phi$, where Φ is an empty set. Lemma 2 also holds.

If layer-*i* does depend on layer-*l* through the layer-layer dependency network $\mathbf{G}$, we will prove Lemma 2 by induction. Let len$(l \rightsquigarrow i)$ be the maximum length of the path from node l to node i on the layer-layer dependency network $\mathbf{G}$. Since $\mathbf{G}$ is a DAG, we have that len$(l \rightsquigarrow i)$ is a finite number.

Base Case. Suppose len$(l \rightsquigarrow i) = 1$, we have that layer-*i* directly depends on layer-*l*. Let $\mathcal{R}_l = \mathcal{J}_l \setminus \mathcal{I}_l$. We have that

$$\mathcal{J}_{l\to i} = \mathcal{I}_{l\to i} \cup \mathcal{R}_{l\to i} \supseteq \mathcal{I}_{l\to i} \tag{16}$$

which complete the proof for the base case where len$(l \rightsquigarrow i) = 1$.

Induction Step. Suppose Lemma 2 holds for len$(l \rightsquigarrow i) \leq q$, where q is a positive integer. We will prove that Lemma 2 also holds for len$(l \rightsquigarrow i) = q + 1$.

Suppose layer-*i* directly depends on layer-i_x $(x = 1, ..., d(i)$, where $d(i)$ is the in-degree of node i on $\mathbf{G}$). Since $\mathbf{G}$ is a DAG, we have that len$(l \rightsquigarrow i_x) \leq q$. By the induction hypothesis, given $\mathcal{I}_l \subseteq \mathcal{J}_l$, we have that $\mathcal{I}_{l\to i_x} \subseteq \mathcal{J}_{l\to i_x}$.

We further have $\mathcal{I}_{l\to i} = \cup_{x=1,\ldots,d(i)} \ (\mathcal{I}_{l\to i_x})_{i_x\to i}$.

Let $\mathcal{R}_{l\to i_x} = \mathcal{J}_{l\to i_x} \setminus \mathcal{I}_{l\to i_x}$ for $x = 1, \ldots, d(i)$. We have that

$$\begin{aligned} \mathcal{J}_{l\to i} &= [\cup_{x=1,\ldots,d(i)} \ (\mathcal{I}_{l\to i_x})_{i_x\to i}] \\ &\cup [\cup_{x=1,\ldots,d(i)} \ (\mathcal{R}_{l\to i_x})_{i_x\to i}] \\ &= \mathcal{I}_{l\to i} \cup \mathcal{R}_{l\to i} \supseteq \mathcal{I}_{l\to i} \end{aligned} \tag{17}$$

which completes the proof of the induction step.

Putting everything together, we have completed the proof for Lemma 2. □

Notice that in the proof of Lemma 2, it requires the layer-layer dependency network $\mathbf{G}$ to be a DAG so that the longest path from the control layer $\mathbf{A}_l$ to any target layer $\mathbf{A}_t$ is of finite length. To further generalize it to arbitrary dependency structures, we need the following lemma, which says that the dependent paths from control layer to target layer in any arbitrarily structured dependency network can be reduced to a DAG.

Lemma 3. DAG Dependency Reduction Lemma. *Given a multi-layered network $\Gamma =< \mathbf{G}, \mathcal{A}, \mathcal{D}, \theta, \varphi >$ with arbitrarily structured layer-layer dependency network $\mathbf{G}$, a control layer $\mathbf{A}_l$, and a target layer $\mathbf{A}_t$, the dependent paths constructed by Algorithm 1 can be reduced to a DAG.*

Proof. In Algorithm 1, Tarjan Algorithm is first used to find out all strongly connected components $\mathcal{V} = \{\mathcal{SC}_1, \mathcal{SC}_2, \ldots, \mathcal{SC}_f\}$ in layer-layer dependency network $\mathbf{G}$. The cross-component dependency edges are denoted as $\mathcal{E} = \{E_{i,j}\}_{i,j=1,\ldots,f,i\neq j}$ where $< u, v >\in E_{i,j}$ iff $\mathbf{G}(u, v) = 1$ and $\mathbf{A}_u \in \mathcal{SC}_i$, $\mathbf{A}_v \in \mathcal{SC}_j$. Based on the node set $\mathcal{V}$ and the edge set $\mathcal{E}$, a directed meta graph $\mathcal{G}$ can be constructed where $\mathcal{G}(u, v) = 1$ iff $E_{i,j} \neq \phi$. The meta graph $\mathcal{G}$ is acyclic. Otherwise, the cycle in $\mathcal{G}$ would be merged into a large strongly connected components by Tarjan Algorithm at the first place. Suppose the control layer $\mathbf{A}_l$ and the target layer $\mathbf{A}_t$ are located in strongly connected component $\mathcal{SC}_i$ and $\mathcal{SC}_j$ respectively, then a set of acyclic paths $\mathcal{P}$ from $\mathcal{SC}_i$ and $\mathcal{SC}_j$ can be extracted from $\mathcal{G}$. To show that the dependent paths from $\mathbf{A}_l$ to $\mathbf{A}_t$ is DAG, we only need to show that each meta path $\mathbf{P} \in \mathcal{P}$ can be unfolded into a DAG.

Here we proceed to show how a meta path $\mathbf{P}$ can be represented with a DAG. As the nodes in $\mathbf{P}$ are strongly connected components that contain cycles, and the edges in $\mathbf{P}$ contain corresponding cross-component edges that would not introduce any cycles, representing $\mathbf{P}$ with a DAG can be converted to a problem of unfolding the cyclic dependent paths in a strongly connected component into a DAG. As described in Algorithm 4, a strongly connected component $\mathcal{Q}$ can be partitioned into two parts: (1) a DAG that contains all acyclic links (denoted as $\mathbf{G}_{\mathcal{Q},0}$) and (2) links that enclose cycles in $\mathcal{Q}$ (denoted as $E_{\mathcal{Q},0}$). Therefore, given a strongly connected component $\mathcal{Q}$ and a set of dependent nodes $\{\mathcal{T}_v\}_{\mathbf{A}_v\in\mathbf{Q}}$ in $\mathcal{Q}$, the dependent cycle can be replaced by a chain of $\mathbf{G}_{\mathcal{Q},0}$'s replicas, where the two adjacent replicas are linked by $E_{\mathcal{Q},0}$ until the number of the dependent nodes in the connected component converges (step 5 to 23 in Algorithm 3). As the number of dependent nodes keeps increasing in each iteration and is upper bounded by the total number of nodes in $\mathcal{Q}$, the repetition is guaranteed to stop at a stable state within finite iterations. Since $\mathbf{G}_{\mathcal{Q},0}$ is a DAG, the links ($E_{\mathcal{Q},0}$) between each replicas $\{\mathbf{G}_{\mathcal{Q},1}, \ldots, \mathbf{G}_{\mathcal{Q},L}\}$ would not introduce any cycle, the resulting graph $\mathbf{G}_{\mathcal{Q}}$ is also a DAG. Therefore, the dependent paths constructed by Algorithm 1 from $\mathbf{A}_l$ and $\mathbf{A}_t$ can be represented as a DAG. □

A complete DAG reduction algorithm is summarized from Algorithm 1 to 4.

Algorithm 1 DAG Reduction Algorithm

```
Input: (1) A multi-layered network $\Gamma$, (2) a control layer $\mathbf{A}_l$, (3)
    a set of node $\mathcal{S}_l$ in layer $\mathbf{A}_l$ and (4) a target layer $\mathbf{A}_t$
Output: (1) a DAG $\mathbf{G}_D$ that contains all the dependent paths
    from $\mathcal{S}_l$ in layer $\mathbf{A}_l$ to $\mathbf{A}_t$ and (2) $\mathcal{S}_{l\to t}$.
 1: find out all strongly connected components in $\mathbf{G}$ as $\mathcal{V} \leftarrow$
    $\{\mathcal{SC}_1, \mathcal{SC}_2, \ldots, \mathcal{SC}_f\}$ with Tarjan Algorithm[2]
 2: set $\mathcal{E} \leftarrow \{E_{i,j}\}_{i,j=1,\ldots,f}$, where $< u, v >\in E_{i,j}$ iff
    $\mathbf{G}(u, v) = 1$ and $\mathbf{A}_u \in \mathcal{SC}_i$, $\mathbf{A}_v \in \mathcal{SC}_j$
 3: construct meta graph $\mathcal{G}$ from $\mathcal{V}$ s.t. $\mathcal{G}(i, j) = 1$ iff $E_{i,j} \neq \phi$
 4: $\mathcal{SC}_i \leftarrow$ connected component that contains $\mathbf{A}_l$
 5: $\mathcal{SC}_j \leftarrow$ connected component that contains $\mathbf{A}_t$
 6: find out all paths $\mathcal{P}$ from $\mathcal{SC}_i$ to $\mathcal{SC}_j$ in $\mathcal{G}$
 7: initialize $\mathbf{G}_D \leftarrow \phi$, $\mathcal{S}_{l\to t} = \phi$
 8: for each path $\mathbf{P}$ in $\mathcal{P}$ do
 9:     $[\mathbf{G}_D^{\mathbf{P}}, \mathcal{S}_{l\to t}^{\mathbf{P}}] \leftarrow unfoldPath(\mathbf{P}, \mathcal{G}, \mathcal{S}_l, \Gamma, \mathcal{V}, \mathcal{E})$
10:     $\mathbf{G}_D \leftarrow \mathbf{G}_D \cup \mathbf{G}_D^{\mathbf{P}}$, $\mathcal{S}_{l\to t} \leftarrow \mathcal{S}_{l\to t} \cup \mathcal{S}_{l\to t}^{\mathbf{P}}$
11: end for
12: return $[\mathbf{G}_D, \mathcal{S}_{l\to t}]$
```

2. A widely used strongly connect component detection algorithm in [12].

In Algorithm 1, step 1 runs Tarjan Algorithm [12] to find out all the strongly connected components in layer-layer dependency network $\mathbf{G}$. Step 2 collects all cross-component edges into set $\mathcal{E}$. In the following step, a meta graph $\mathcal{G}$ is constructed based on $\mathcal{V}$ and $\mathcal{E}$. In step 4 and 5, the connected components that contain control layer and target layer are located ($\mathcal{SC}_i$ and $\mathcal{SC}_j$). Step 6 finds out all meta paths from $\mathcal{SC}_i$ to $\mathcal{SC}_j$. In step 7, the final DAG $\mathbf{G}_D$ and dependent node set $\mathcal{S}_{l\to t}$ are initialized as empty sets. From step 8 to step 11, the DAG $\mathbf{G}_D^{\mathbf{P}}$ and dependent node set $\mathcal{S}_{l\to t}^{\mathbf{P}}$ for each path $\mathbf{P}$ in $\mathcal{P}$ are calculated by function $unfoldPath$, and are used to update $\mathbf{G}_D$ and $\mathcal{S}_{l\to t}$ in step 10.

To illustrate how Algorithm 1 works, we present a simple example in Fig. 3. In the example, the dependency network $\mathbf{G}$ contains three layers, where $\mathbf{A}_1$ is the control layer and $\mathbf{A}_3$ is the target layer. Specifically, $\mathbf{A}_2$ is a dependent layer of $\mathbf{A}_1$; while $\mathbf{A}_2$ and $\mathbf{A}_3$ are inter-dependent to each other. The toy example has two strongly connected components $\{\mathcal{SC}_1, \mathcal{SC}_2\}$ and one cross-component edge set $E_{1,2} = \{<1,2>\}$. The meta graph $\mathcal{G}$ is a link graph with just two nodes.

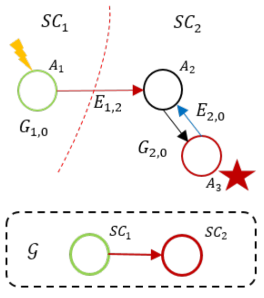


Fig. 3. A cyclic dependency multi-layered network.

In Algorithm 2, the first connected component $\mathcal{Q}$ is initialized as the connected component that contains control layer $\mathbf{A}_l$ in step 2, the dependent nodes are initialized as $\mathcal{S}_l$ from step 5 to 8 and the root layer $\mathbf{R}$ is initialized as the control layer $\mathbf{A}_l$. From step 10 to 36, the DAG $\mathbf{G}_{\mathcal{Q}}^{\mathbf{P}}$ and the final dependent nodes in $\mathcal{Q}$ are calculated by function $unfoldSC$ in line 11; $\mathbf{G}_{\mathcal{Q}}^{\mathbf{P}}$ is then added to the final DAG $\mathbf{G}_D^{\mathbf{P}}$ via cross-component links E_{i_q'',i_q} from step 15 to 17. The initial dependent nodes for the next connected component $\mathcal{SC}_{i_q'}$ are computed through cross-component links $E_{i_q,i_q'}$ from step 27 to 29. Step 30 is used to pick a root layer with non-empty dependent node set for $\mathcal{SC}_{i_q'}$.

Algorithm 3 is used to unfold a strongly connected component into a DAG. In step 1, the input connected component $\mathcal{Q}$ is partitioned into a DAG $\mathbf{G}_{\mathcal{Q},0}$ and a set of cycle links $E_{\mathcal{Q},0}$. In step 2, the DAG $\mathbf{G}_{\mathcal{Q}}$ is initialized by $\mathbf{G}_{\mathcal{Q},1}$, which is a replica of $\mathbf{G}_{\mathcal{Q},0}$. From step 5 to 23, the algorithm keeps appending replicas of $\mathbf{G}_{\mathcal{Q},0}$ ($\mathbf{G}_{\mathcal{Q},c+1}$) onto $\mathbf{G}_{\mathcal{Q}}$ (step 8 to 16) until no new nodes are added to the dependent node set $\{\mathcal{T}_v\}_{\mathbf{A}_v\in\mathcal{Q}}$ (step 17-19).

For the example in Fig. 3, $\mathcal{SC}_1$ is unfolded as $\mathbf{G}_1$ with one node $\mathbf{A}_1^1$ in Fig. 4. The initial dependent node set $\mathcal{T}_2$ for layer $\mathbf{A}_2$ can be calculated through $E_{1,2}$ as $\mathcal{T}_{1\to 2}$. For $\mathcal{SC}_2$, it is first partitioned into a DAG $\mathbf{G}_{2,0}$ and a cycle edge set $E_{2,0} = \{<\mathbf{A}_3, \mathbf{A}_2>\}$ as shown in Fig. 3. Suppose that the dependent node set in $\mathcal{SC}_2$ converges in L_2 iterations, then the DAG for $\mathcal{SC}_2$ can be presented with L_2 replicas of $\mathbf{G}_{2,0}$ linked by edges $\{<\mathbf{A}_3^c, \mathbf{A}_2^{c+1}>\}_{c=1,\ldots,L_2-1}$ as shown in Fig. 4. Putting all together, the final DAG $\mathbf{G}_D$ can be constructed by linking $\mathbf{A}_1^1$ in $\mathbf{G}_1$ with $\{\mathbf{A}_2^x\}_{x=1,\ldots,L_2}$ in $\mathbf{G}_2$.

Algorithm 4 is used to partition a strongly connected component $\mathcal{Q}$ into a DAG $\mathbf{G}_{\mathcal{Q}}$ and an edge set $E_{\mathcal{Q},0}$ that contains all cycle edges. The basic idea is to use Breadth-First-Search algorithm to traverse all the edges in the graph. In step 1 and 2, $\mathbf{G}_{\mathcal{Q},0}$ and $E_{\mathcal{Q},0}$ are initialized as $\mathcal{Q}$ and ϕ respectively. For

Algorithm 2 UnfoldPath: Construct DAG from meta path

Input: (1) A meta path $\mathbf{P} = \mathcal{SC}_i \to \ldots \to \mathcal{SC}_j$, (2) a meta graph $\mathcal{G}$, (3) a set of nodes $\mathcal{S}_l$ in $\mathbf{A}_l \in \mathcal{SC}_i$, (4) a multi-layered network Γ, (5) all strongly connected components $\mathcal{V}$ and (6) all cross-component edges $\mathcal{E}$

Output: (1) a DAG $\mathbf{G}_D^{\mathbf{P}}$ and (2) $\mathcal{S}_{l\to t}^{\mathbf{P}}$.

```
1:  append $\phi$ to the end of meta path $\mathbf{P}$
2:  set $\mathcal{Q} = \mathcal{SC}_i$
3:  $i_q \leftarrow$ index of connected component $\mathcal{Q}$ in meta graph $\mathcal{G}$
4:  $i_q'' \leftarrow -1$
5:  for each layer $\mathbf{A}_v$ in $\mathcal{Q}$ do
6:    initialize $\mathcal{T}_v \leftarrow \phi$
7:  end for
8:  $\mathcal{T}_l \leftarrow \mathcal{T}_l \cup \mathcal{S}_l$
9:  set root $\mathbf{R} \leftarrow \mathbf{A}_l$
10: while true do
11:   $[\mathbf{G}_{\mathcal{Q}}^{\mathbf{P}}, \{\mathcal{S}_{l\to v}^{\mathbf{P}}\}_{\mathbf{A}_v\in\mathcal{Q}}] \leftarrow$
          $unfoldSC(\mathcal{Q}, \{\mathcal{T}_v\}_{\mathbf{A}_v\in\mathcal{Q}}, \mathbf{R})$
12:   if $i_q'' = -1$ then
13:     $\mathbf{G}_D^{\mathbf{P}} \leftarrow \mathbf{G}_{\mathcal{Q}}^{\mathbf{P}}$
14:   else
15:     for each $<u, v> \in E_{i_q'',i_q}$ do
16:       link layer $\mathbf{A}_u^{L_{i_q''}} \in \mathbf{G}_D^{\mathbf{P}}$ to layers $\{\mathbf{A}_v^x\}_{x=1,\ldots,L_{i_q}} \in \mathbf{G}_{\mathcal{Q}}^{\mathbf{P}}$
17:     end for
18:   end if
19:   $\mathcal{Q}' \leftarrow \mathcal{Q}.successor()$
20:   if $\mathcal{Q}' = \phi$ then
21:     break
22:   else
23:     $i_q' \leftarrow$ index of $\mathcal{Q}'$ in meta graph $\mathcal{G}$
24:     for each layer $\mathbf{A}_v$ in $\mathcal{Q}'$ do
25:       initialize $\mathcal{T}_v \leftarrow \phi$
26:     end for
27:     for each edge $<u, v> \in E_{i_q,i_q'}$ do
28:       $\mathcal{T}_v \leftarrow \mathcal{T}_v \cup (\mathcal{S}_{l\to u}^{\mathbf{P}})_{u\to v}$
29:     end for
30:     $\mathbf{R} \leftarrow \mathbf{A}_r$, where $\mathbf{A}_r$ is a randomly picked layer from $\mathcal{Q}'$ with $\mathcal{T}_r \neq \phi$
31:     $\mathcal{Q} \leftarrow \mathcal{Q}'$
32:     $i_q'' \leftarrow i_q$
33:     $i_q \leftarrow i_q'$
34:   end if
35: end while
36: return $\mathbf{G}_D^{\mathbf{P}}, \mathcal{S}_{l\to t}^{\mathbf{P}}$
```

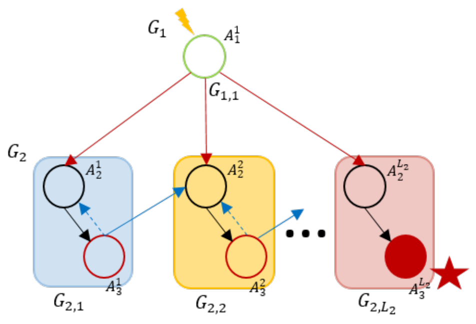


Fig. 4. Constructed DAG for Fig. 3.

each edge $<\mathbf{A}_u, \mathbf{A}_v>$ in $\mathcal{Q}$, if $\mathbf{A}_v$ appears in $\mathbf{A}_u$'s ancestor list $\mathcal{L}_u$, then $<\mathbf{A}_u, \mathbf{A}_v>$ would be removed from $\mathbf{G}_{\mathcal{Q},0}$ and added

Algorithm 3 UnfoldSC: Construct DAG from strongly connected component

```
Input: (1) A strongly connected component $\mathcal{Q}$, (2) a set of initial nodes for each layer $\{\mathcal{T}_v\}_{\mathbf{A}_v \in \mathbf{Q}}$, (3) a root layer $\mathbf{R}$
Output: (1) a DAG $\mathbf{G}_{\mathcal{Q}}$ (2) $\{\mathcal{S}_{l\to v}\}_{\mathbf{A}_v \in \mathcal{Q}}$.
1: extract DAG and cycle edges $[\mathbf{G}_{\mathcal{Q},0}, E_{\mathcal{Q},0}] \leftarrow extractDAG(\mathcal{Q}, \mathbf{R})$
2: set $\mathbf{G}_{\mathcal{Q},1} \leftarrow \mathbf{G}_{\mathcal{Q},0}$, denote the layers in $\mathbf{G}_{\mathcal{Q},1}$ as $\{\mathbf{A}_v^1\}$
3: set $c \leftarrow 1$
4: initialize $\mathbf{G}_{\mathcal{Q}} \leftarrow \mathbf{G}_{\mathcal{Q},1}$
5: while true do
6:   $\{\mathcal{T}_v^c\}_{\mathbf{A}_v \in \mathcal{Q}} \leftarrow$ dependents of $\{\mathcal{T}_v\}_{\mathbf{A}_v \in \mathcal{Q}}$ in $\mathbf{G}_{\mathcal{Q},c}$
7:   update $\{\mathcal{T}_v\}_{\mathbf{A}_v \in \mathcal{Q}} \leftarrow \{\mathcal{T}_v \cup \mathcal{T}_v^c\}_{\mathbf{A}_v \in \mathcal{Q}}$
8:   set $\mathbf{G}_{\mathcal{Q},c+1} \leftarrow \mathbf{G}_{\mathcal{Q},0}$, layers in $\mathbf{G}_{\mathcal{Q},c+1}$ are denoted as $\{\mathbf{A}_v^{c+1}\}$
9:   extend $\mathbf{G}_{\mathcal{Q}} \leftarrow \mathbf{G}_{\mathcal{Q}} \cup \mathbf{G}_{\mathcal{Q},c+1}$
10:  for each edge $< u, v > \in E_{\mathcal{Q},0}$ do
11:    $\mathcal{T}_{u\to v} \leftarrow$ all dependents of $\mathcal{T}_u$ in layer $\mathbf{A}_v$
12:    if $\mathcal{T}_{u\to v} \nsubseteq \mathcal{T}_v$ then
13:      add edge $< \mathbf{A}_u^c, \mathbf{A}_u^{c+1} >$ to $\mathbf{G}_{\mathcal{Q}}$
14:      update $\mathcal{T}_v \leftarrow \mathcal{T}_v \cup \mathcal{T}_{u\to v}$
15:    end if
16:  end for
17:  if no edge added between $\mathbf{G}_{\mathcal{Q},c}$ and $\mathbf{G}_{\mathcal{Q},c+1}$ then
18:    remove $\mathbf{G}_{\mathcal{Q},c+1}$ from $\mathbf{G}_{\mathcal{Q}}$
19:    break
20:  else
21:    $c \leftarrow c + 1$
22:  end if
23: end while
24: return $[\mathbf{G}_{\mathcal{Q}}, \{\mathcal{T}_v\}_{\mathbf{A}_v \in \mathcal{Q}}]$
```

Algorithm 4 ExtractDAG: extract DAG from strongly connected component

```
Input: (1) A strongly connected component $\mathcal{Q}$ and (2) a root layer $\mathbf{R}$ in the connected component
Output: (1) a DAG $\mathbf{G}_{\mathcal{Q},0}$ (2) edge set $E_{\mathcal{Q},0}$ that contains all cycle edges.
1: initialized $\mathbf{G}_{\mathcal{Q},0} \leftarrow \mathcal{Q}$
2: initialized $E_{\mathcal{Q},0} \leftarrow \phi$
3: for each layer $\mathbf{A}_v \in \mathcal{Q}$ do
4:   initialize its ancestor list $\mathcal{L}_v \leftarrow \phi$
5: end for
6: initialize a queue $\mathcal{T} \leftarrow \phi$
7: $\mathcal{T}.enqueue(\mathbf{R})$
8: while $\mathcal{T} \neq \phi$ do
9:   $\mathbf{A}_u \leftarrow \mathcal{T}.dequeue()$
10:  for each dependent layer $\mathbf{A}_v$ of $\mathbf{A}$ do
11:    if $\mathbf{A}_v \in \mathcal{L}_u$ then
12:      remove edge $< \mathbf{A}_u, \mathbf{A}_v >$ from $\mathbf{G}_{\mathcal{Q},0}$
13:      $E_{\mathcal{Q},0} \leftarrow E_{\mathcal{Q},0} \cup < \mathbf{A}_u, \mathbf{A}_v >$
14:    else
15:      $\mathcal{T}.enqueue(\mathbf{A}_v)$
16:      $\mathcal{L}_v \leftarrow \mathcal{L}_v \cup \mathcal{L}_u \cup \{\mathbf{A}_u\}$
17:    end if
18:  end for
19: end while
20: return $[\mathbf{G}_{\mathcal{Q},0}, E_{\mathcal{Q},0}]$
```

to $E_{\mathcal{Q},0}$ (step11 to 13).

The algorithms used in Lemma 3 together with Lemma 2 guarantee that set-ordering preservation property also holds in multi-layered networks with arbitrarily structured dependency graph $\mathbf{G}$.

Now, we are ready to present our main theorem as follows.

Theorem 1. Diminishing Returns Property of MULAN. *For* **any** *connectivity function $C(\mathbf{A})$ in the* SUBLINE *family (eq. (1)) and* **any** *multi-layered network in the* MULAN *family (Definition 1); the overall impact of node set $\mathcal{S}_l$ in the control layer l, $\mathbb{I}(\mathcal{S}_l) = \sum_{i=1}^{g} \alpha_i I(\mathcal{S}_{l\to i})$, is (a) monotonically non-decreasing; (b) sub-modular; and (c) normalized.*

Proof. We first prove the sub-modularity of function $\mathbb{I}(\mathcal{S}_l)$. Let $\mathcal{I}_l, \mathcal{J}_l, \mathcal{K}_l$ be three node sets in layer $\mathbf{A}_l$ and $\mathcal{I}_l \subseteq \mathcal{J}_l$. Define the following two sets as: $\mathcal{S}_l = \mathcal{I}_l \cup \mathcal{K}_l$ and $\mathcal{T}_l = \mathcal{J}_l \cup \mathcal{K}_l$. We have that

$$\begin{aligned}\mathbb{I}(\mathcal{S}_l) - \mathbb{I}(\mathcal{I}_l) &= \sum_{i=1}^{g} \alpha_i I(\mathcal{S}_{l\to i}) - \sum_{i=1}^{g} \alpha_i I(\mathcal{I}_{l\to i}) \\ &= \sum_{i=1}^{g} \alpha_i (I(\mathcal{S}_{l\to i}) - I(\mathcal{I}_{l\to i}))\end{aligned} \tag{18}$$

$$\begin{aligned}\mathbb{I}(\mathcal{T}_l) - \mathbb{I}(\mathcal{J}_l) &= \sum_{i=1}^{g} \alpha_i I(\mathcal{T}_{l\to i}) - \sum_{i=1}^{g} \alpha_i I(\mathcal{J}_{l\to i}) \\ &= \sum_{i=1}^{g} \alpha_i (I(\mathcal{T}_{l\to i}) - I(\mathcal{J}_{l\to i}))\end{aligned} \tag{19}$$

$\forall i = 1, \ldots, g$, it is obvious that $\mathcal{S}_{l\to i} = \mathcal{I}_{l\to i} \cup \mathcal{K}_{l\to i}$, $\mathcal{T}_{l\to i} = \mathcal{J}_{l\to i} \cup \mathcal{K}_{l\to i}$. By Lemma 2, we have $\mathcal{I}_{l\to i} \subseteq \mathcal{J}_{l\to i}$. Furthermore, by the sub-modularity of $I(\mathcal{S}_i)$ on $\mathbf{A}_i$ (Lemma 1), we have that

$$I(\mathcal{S}_{l\to i}) - I(\mathcal{I}_{l\to i}) \geq I(\mathcal{T}_{l\to i}) - I(\mathcal{J}_{l\to i})$$

Since for $\forall i$, we have $\alpha_i \geq 0$. Therefore

$$\sum_{i=1}^{g} \alpha_i (I(\mathcal{S}_{l\to i}) - I(\mathcal{I}_{l\to i})) \geq \sum_{i=1}^{g} \alpha_i (I(\mathcal{T}_{l\to i}) - I(\mathcal{J}_{l\to i})) \tag{20}$$

Putting eq. (18), (19) and (20) together, we have that

$$\mathbb{I}(\mathcal{S}_l) - \mathbb{I}(\mathcal{I}_l) \geq \mathbb{I}(\mathcal{T}_l) - \mathbb{I}(\mathcal{J}_l)$$

which completes the proof that $\mathbb{I}(\mathcal{S}_l)$ is sub-modular.

Notice that the connectivity function $C(\mathbf{A})$ in the SUBLINE family is monotonically non-decreasing. By eq. (18), we have that

$$\mathbb{I}(\mathcal{S}_l) - \mathbb{I}(\mathcal{I}_l) = \sum_{i=1}^{g} \alpha_i (C(\mathbf{A}_i \setminus \mathcal{I}_l) - C(\mathbf{A}_i \setminus \mathcal{S}_l)) \geq 0$$

which completes the proof that $\mathbb{I}(\mathcal{S}_l)$ is monotonically non-decreasing.

Finally, notice that for each dependent layer, the impact function $I(\mathcal{S}_i)$ is normalized (Lemma 1); and for $i = 1, \ldots, g$, $\Phi_{l\to i} = \Phi$ (an empty set). Therefore we have that $\mathbb{I}(\Phi) = 0$. In other words, $\mathbb{I}(\mathcal{S}_l)$ is also normalized. □

4.3 OPERA: Algorithms and Analysis

In this subsection, we introduce our algorithm to solve OPERA (Problem 1), followed by some analysis in terms of the optimization quality as well as the complexity.

A Generic Solution Framework. Finding out the global optimal solution for Problem 1 by a brute-force method would be computationally intractable, due to the exponential enumeration. Nonetheless, the diminishing returns property of the impact function $\mathbb{I}(.)$ (Theorem 1) immediately lends itself to a greedy algorithm for solving OPERA with any connectivity function in

Algorithm 5 OPERA: A Generic Solution Framework

```
Input: (1) A multi-layered network $\Gamma$, (2) a control layer $\mathbf{A}_l$, (3) an overall impact function $\mathbb{I}(\mathcal{S}_l)$ and (4) an integer $k$
Output: a set of $k$ nodes $\mathcal{S}$ from the control layer $\mathbf{A}_l$.
1: initialize $\mathcal{S}$ to be empty
2: for each node $v_0$ in layer $\mathbf{A}_l$ do
3:   calculate margin$(v_0) \leftarrow \mathbb{I}(v_0)$
4: end for
5: find $v = \text{argmax}_{v_0}\text{margin}(v_0)$ and add $v$ to $\mathcal{S}$
6: set margin$(v) \leftarrow -1$
7: for $i = 2$ to $k$ do
8:   set maxMargin $\leftarrow -1$
9:   for each node $v_0$ in layer $\mathbf{A}_l$ do
10:    /*an optional 'if' for lazy eval.*/
11:    if margin$(v_0) >$ maxMargin then
12:      calculate margin$(v_0) \leftarrow \mathbb{I}(\mathcal{S} \cup \{v_0\}) - \mathbb{I}(\mathcal{S})$
13:      if margin$(v_0) >$ maxMargin then
14:        set maxMargin $\leftarrow$ margin$(v_0)$ and $v \leftarrow v_0$
15:      end if
16:    end if
17:  end for
18:  add $v$ to $\mathcal{S}$ and set margin$(v) \leftarrow -1$
19: end for
20: return $\mathcal{S}$
```

the SUBLINE family and arbitrary member in the MULAN family, as summarized in Algorithm 5.

In Algorithm 5, step 2-4 calculate the impact score $\mathbb{I}(v_0)$ $(v_0 = 1, 2, ...)$ for each node in the control layer $\mathbf{A}_l$. Step 5 selects the node with the maximum impact score. In each iteration in step 7-19, we select one of the remaining $(k-1)$ nodes, which would make the maximum marginal increase in terms of the current impact score (step 12, margin$(v_0) = \mathbb{I}(\mathcal{S} \cup \{v_0\}) - \mathbb{I}(\mathcal{S})$). In order to further speed-up the computation, the algorithm admits an *optional* lazy evaluation strategy (adopted from [13]) by activating an optional 'if' condition in Step 11.

Note that it is easy to extend Algorithm 5 to the scenario where we have multiple control layers. Suppose $\mathcal{A}_l = \{\mathbf{A}_{l_1}, \mathbf{A}_{l_2}, \ldots, \mathbf{A}_{l_x}\}$ is a set of control layers, to select best k nodes from $\mathcal{A}_l$, we only need to scan over all the nodes in $\mathcal{A}_l$ in step 2 and step 9 respectively, and pick the highest impact node from the entire candidate set in step 5 and 18. Consequently, the resulting set $\mathcal{S}$ returned from the algorithm would contain the k highest impact nodes over $\mathcal{A}_l$.

Algorithm Analysis. Here, we analyze the optimality as well as the complexity of Algorithm 5, which are summarized in Lemma 4-6. According to these lemmas, our proposed Algorithm 1 leads to a *near-optimal* solution with a *linear* complexity.

Lemma 4. Near-optimality. *Let $\mathcal{S}_l$ and $\tilde{\mathcal{S}}_l$ be the sets selected by Algorithm 5 and the brute-force algorithm, respectively. Let $\mathbb{I}(\mathcal{S}_l)$ and $\mathbb{I}(\tilde{\mathcal{S}}_l)$ be the overall impact of $\mathcal{S}_l$ and $\tilde{\mathcal{S}}_l$. Then $\mathbb{I}(\mathcal{S}_l) \geq (1-1/e)\mathbb{I}(\tilde{\mathcal{S}}_l)$.*

Proof. As proved in Theorem 1, the overall impact function $\mathbb{I}(\mathcal{S})$ $(\mathcal{S} \subseteq \mathbf{A}_l)$ is monotonic, sub-modular and normalized. Using the property of such functions in [14], we have $\mathbb{I}(\mathcal{S}_l) \geq (1-1/e)\mathbb{I}(\tilde{\mathcal{S}}_l)$. □

Lemma 5. Time complexity. *Let $h(n_i, m_i, |\mathcal{S}_{l\to i}|)$ be the time to compute the impact of node set $\mathcal{S}_l$ on layer i. The time complexity for selecting $\mathcal{S}$ of size k from the control layer $\mathbf{A}_l$ is upper bounded by $O(k(|\mathcal{N}(\mathbf{A}_l)| + n_l \sum_{i=1}^{g} h(n_i, m_i, |\mathcal{S}_{l\to i}|)))$ where $\mathcal{N}(\mathbf{A}_l)$ denotes the nodes and cross-layer links in Γ that depends on $\mathbf{A}_l$.*

Proof. The greedy algorithm with lazy evaluation strategy needs to iterate over all the nodes in layer $\mathbf{A}_l$ for k time. At each time, the worst case is that we need to evaluate the marginal increase for all unselected nodes in $\mathbf{A}_l$. The overall complexity of finding dependents of every nodes in $\mathbf{A}_l$ is equal to the size of the sub-system that rooted on $\mathbf{A}_l$, which is $|\mathcal{N}(\mathbf{A}_l)|$. And for each unselected node, finding out its current impact to the system as shown in step 3 and step 12 can be upper bounded by the complexity of $\sum_{i=1}^{g} h(n_i, m_i, n_i) + g$ because there are at most g non-zero weighted layers that depends on $\mathbf{A}_l$. Taking these all together, the complexity of selecting set $\mathcal{S}$ from $\mathbf{A}_l$ with Algorithm 5 is $O(k[|\mathcal{N}(\mathbf{A}_l)| + n_l \sum_{i=1}^{g} h(n_i, m_i, |\mathcal{S}_{l\to i}|)])$, where $|\mathcal{N}(\mathbf{A}_l)|$ is upper bounded by $N + L$, which is the sum of total number of nodes and total number of dependency links in Γ. If given that function h is linear to n_i, m_i and $|\mathcal{S}_{l\to i}|$, as $|\mathcal{S}_{l\to i}|$ is upper bounded by n_i, and n_l can be viewed as a constant compared to N, M and L, it is easy to see that the complexity of the algorithm is linear to N, M and L. □

Remarks. Lemma 5 implies a linear time complexity of the proposed OPERA algorithm wrt the size of the entire multi-layered network $(N + M + L)$, where N, M, L are the total number of nodes, the total number of within-layer links and the total number of cross-layer links in Γ under the condition that the function h is linear wrt n_i, m_i and $|\mathcal{S}_{l\to i}|$. This condition holds for most of the network connectivity measures in the SUBLINE family, e.g., the path capacity, the truncated loop capacity and the triangle capacity. To see this, let us take the most expensive truncated loop capacity as an example. The time complexity for calculating truncated loop capacity in a single network is $O(mr + nr^2)$, where r is the number of eigenvalues used in the calculation and it is often orders of magnitude smaller compared with m and n. On the other hand, we have $|\mathcal{N}(\mathbf{A}_l)| \leq N + L$. Therefore, the overall time complexity for selecting set $\mathcal{S}$ of size k from control layer $\mathbf{A}_l$ is upper bounded by $O(k(N + L + n_l \sum_{i=1}^{g}(m_i r + n_i r^2))) = O(k(N + L + n_l(rM + r^2N))) = O(k(L + n_l(rM + r^2N)))$.

Lemma 6. Space complexity. *Let $w(n_i, m_i, |\mathcal{S}_{l\to i}|)$ be a function of n_i, m_i and $|\mathcal{S}_{l\to i}|$ that denotes the space cost to compute $I(\mathcal{S}_{l\to i})$. The space complexity of Algorithm 5 is $O(N + M + L)$ under the condition that the function w is linear wrt n_i, m_i and $|\mathcal{S}_{l\to i}|$.*

Proof. As defined in 5, N, M, L are the total number of nodes, total number of within-layer links and total number of cross-layer links in Γ. Then storing multi-layered network Γ would take a space of $O(N+M+L)$. In Algorithm 5, it takes $O(n_l)$ to save the marginal increase vector (*margin*) and $O(k)$ to save result $\mathcal{S}$. As space for computing $I(\mathcal{S}_{l\to i})$ can be reused for each layer i, then computing $\mathbb{I}(\mathcal{S}_{l\to i})$ is bounded by $\arg max_i w(n_i, m_i, |\mathcal{S}_{l\to i}|)$. If function w is linear wrt n_i, m_i and $|\mathcal{S}_{l\to i}|$, then the space complexity of Algorithm 5 is of $O(N + M + L + k + n_l) + O(\arg max_i(n_i)) + O(\arg max_i(m_i)) = O(N + M + L)$. □

Remarks. The condition that the function w is linear wrt n_i, m_i and $|\mathcal{S}_{l\to i}|$ holds for most of the network connectivity measures in the SUBLINE family, which in turn implies a linear space complexity for the proposed OPERA algorithm. Again, let us take the truncated loop capacity connectivity as an example. Storing the input MULAN (Γ) takes $O(N + M + L)$ in space. The space cost to calculate the truncated loop capacity in a single-layered network is $O(m + nr)$, where r is the number of eigenvalues used for the computation. Again, r is usually a much smaller number compared with m and n, and thus is considered as a constant. Therefore, the overall space complexity for OPERA with the truncated loop capacity is $O(N + M + L)$.

5 EXPERIMENTAL RESULTS

In this section, we empirically evaluate the proposed OPERA algorithms. All experiments are designed to answer the following two questions:

- *Effectiveness*: how effective are the proposed OPERA algorithms at optimizing the connectivity measures (defined in the proposed SUBLINE family) of a multi-layered network (from the proposed MULAN family)?
- *Efficiency*: how fast and scalable are our algorithms?

5.1 Experimental Setup

Data Sets Summary. We perform the evaluations on four application domains, including (D1) a multi-layered Internet topology at the autonomous system level (MULTIAS); (D2) critical infrastructure networks (INFRANET); (D3) a social-information collaboration network (SOCINNET); and (D4) a biological CTD (Comparative Toxicogenomics Database) network [15] (BIO). For the first two domains, we use real networks to construct the within-layer networks (i.e., $\mathcal{A}$ in the MULAN model) and construct one or more cross-layer dependency structures based on real application scenarios (i.e., $\mathbf{G}$ and $\mathcal{D}$ in the MULAN model). For the data sets in SOCINNET and BIO domains, both the within-layer networks and cross-layer dependency networks are based on real connections. A summary of these data sets is shown in Table 2. We will present the detailed description of each application domains in Subsection 5.2.

TABLE 2
Data Sets Summary.

Data Sets	Application Domains	# of Layers	# of Nodes	# of Links
D1	MULTIAS	2∼4	5,929∼24,539	11,183∼50,778
D2	INFRANET	3	19,235	46,926
D3	SOCINNET	2	63,501∼124,445	13,097∼211,776
D4	BIO	3	35,631	253,827

Baseline Methods. To our best knowledge, there is no existing method which can be directly applied to the connectivity optimization problem (Problem 1) of the MULAN model. We generate the baseline methods using two complementary strategies, including *forward propagation* ('FP' for short) and *backward propagation* ('BP' for short). The key idea behind the *forward propagation* strategy is that an important node in *control layer* might have more impact on its dependent networks as well. On the other hand, for the *backward propagation strategy*, we first identify important nodes in the *target layer(s)*, and then trace back to its supporting layer(s) through the cross-layer dependency links (i.e., $\mathcal{D}$). For both strategies, we need a node importance measure. In our evaluations, we compare three different measures, including (1) node degree; (2) pagerank measure [16]; and (3) *Netshield* values [8]. In addition, for comparison purpose, we also randomly select nodes either from the control layer (for the forward propagation strategy) or from the target layer(s) (for the backward propagation strategy). Altogether, we have eight baseline methods (four for each strategy, respectively), including (1) 'Degree-FP', (2) 'PageRank-FP', (3) 'Netshield-FP', (4) 'Rand-FP', (5) 'Degree-BP', (6) 'PageRank-BP', (7) 'Netshield-BP', (8) 'Rand-BP'.

OPERA *Algorithms and Variants*. We evaluate three prevalent network connectivity measures, including (1) the leading eigenvalue of the (within-layer) adjacency matrix, which relates to the epidemic threshold of a variety of cascading models; (2) the loop capacity (LC), which relates to the robustness of the network; and (3) the triangle capacity (TC), which relates to the local connectivity of the network. As mentioned in Section 3, both the loop capacity and the triangle capacity are members of the SUBLINE family. Strictly speaking, the leading eigenvalue does *not* belong to the SUBLINE family. Instead, it approximates the path capacity (PC), and the latter (PC) is a member of the SUBLINE family. Correspondingly, we have three instances of the proposed OPERA algorithm (each corresponding to one specific connectivity measures) including 'OPERA-PC', 'OPERA-LC', and 'OPERA-TC'. Recall that there is an optional lazy evaluation step (step 11) in the proposed OPERA algorithm, thanks to the diminishing returns property of the SUBLINE connectivity measures. When the leading eigenvalue is chosen as the connectivity function, such diminishing returns property does not hold any more. To address this issue, we introduce a variant of OPERA-PC as follows. At each iteration, after the algorithm chooses a new node v (step 18, Algorithm 1), we (1) update the network by removing all the nodes that depend on node v, and (2) update the corresponding leading eigenvalues and eigenvectors. We refer to this variant as 'OPERA-PC-Up'. For each of the three connectivity measures, we run all four OPERA algorithms.

Machines and Repeatability. All the experiments are performed on a machine with 2 processors Intel Xeon 3.5GHz with 256GB of RAM. The algorithms are programmed with MATLAB using single thread. All the data sets used in this paper are publicly available. We will open source all the codes after the paper is accepted.

5.2 Effectiveness Results

D1 - MULTIAS. This data set contains the Internet topology at the autonomous system level. The data set is available at http://snap.stanford.edu/data/. It has 9 different network snapshots, with $633 \sim 13,947$ nodes and $1,086 \sim 30,584$ edges. In our evaluations, we treat these snapshots as the within-layer adjacency matrices $\mathcal{A}$. For a given supporting layer, we generate the cross-layer node-node dependency matrices $\mathcal{D}$ by randomly choosing 3 nodes from its dependent layer as the direct dependents for each supporting node. For this application domain, we have experimented with different layer-layer dependency structures ($\mathbf{G}$), including a three-layered line-structured network, a three-layered tree-structured network, a four-layered diamond shaped network and a three-layered cyclic network. As the experimental results in the first three networks follows similar pattern, we only present the results on diamond shaped network and cyclic network in Fig. 5 and 6 due to page limits. Overall, the four instances of the proposed OPERA algorithm perform better than the baseline methods. Among the baseline methods, the *backward propagation* methods are better than the forward propagation methods under acyclic dependency networks (5). This is because the length of the back tracking path on the dependency network $\mathbf{G}$ (from the target layer to the control layer) is short. Therefore, compared with other baseline methods, the node set returned from the BP strategy is able to affect more important nodes in the target layer. While for the cyclic dependency network in Fig. 6, the back tracking path is elongated by the cycle. Then the nodes selected by BP strategy are not guaranteed to affect more important nodes in the target layer than FP strategy.

D2 - INFRANET. This data set contains three types of critical infrastructure networks, including (1) the power grid, (2) the communication network; and (3) the airport networks. The power grid is an undirected, un-weighted network representing the topology of the Western States Power Grid of the United State [17]. It has 4,941 nodes and 6,594 edges. We use one snapshot from the MULTIAS data set as the communication network with 11,461

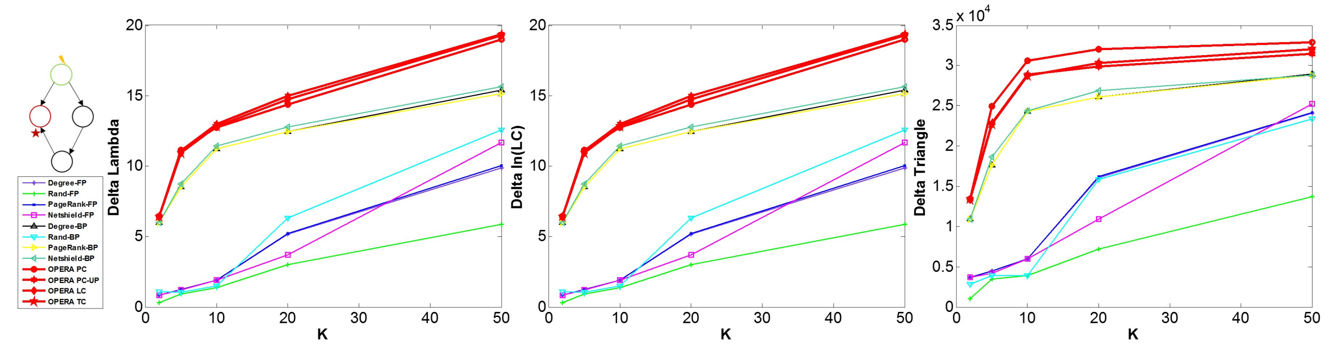


Fig. 5. Evaluations on the MULTIAS data set, with a four-layered diamond-shaped dependency network. The connectivity change vs. budget. Larger is better. All the four instances of the proposed OPERA algorithm (in red) outperform the baseline methods.

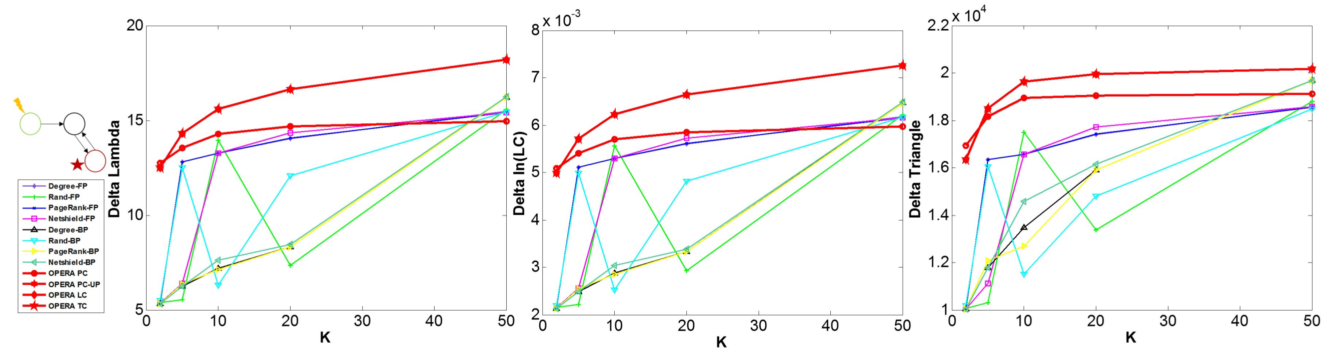


Fig. 6. Evaluations on the MULTIAS data set, with a three-layered cyclic dependency network. The connectivity change vs. budget. Larger is better. Three out of four instances of the proposed OPERA algorithm (in red) outperform the baseline methods.

nodes and 32,730 edges. The airport network represents the internal US air traffic lines between 2,649 airports and has 13,106 links (available at http://www.levmuchnik.net/Content/Networks/NetworkData.html). We construct a triangle-shaped layer-layer dependency network $\mathbf{G}$ (see the icon of Fig. 7) based on the following observation. The operation of an airport depends on both the electricity provided by the power grid and the Internet support provided by the communication network. In the meanwhile, the full-functioning of the communication network depends on the support of power grid. We use similar strategy as in MULTIAS to generate the cross-layer node-node dependency matrices $\mathcal{D}$. The results are summarized in Fig. 7. Again, the proposed OPERA algorithms outperform all the baseline methods. Similar to the MULTIAS network, the back tracking path from the airport layer to the power grid layer is also very short. Therefore, the backward propagation strategies perform relatively better than other baseline methods. In addition, we change the density of the cross-layer node-node dependency matrices and evaluate its impact on the optimization results in Fig. 8. We found that (1) across different dependency densities, the proposed OPERA algorithms still outperform the baseline methods; and (2) when the dependency density increases, the algorithms lead to a larger decrease of the corresponding connectivity measures with the same budget.

D3 - SOCINNET. This data set contains three types of social-information networks [18], including (1) a co-authorship network; (2) a paper-paper citation network; and (3) a venue-venue citation network. Different from the previous two data sets, two types of cross-layer node-node dependency links naturally exist in this data set, including *who-writes-which paper*, and *which venue-publishes-which paper*. In our experiment, we use the papers published between year 1990 to 1992. In total, there are 62,602 papers, 61,843 authors, 899 venues, 10,739 citation links, 201,037 collaboration links, 2,358 venue citation links, 126,242 author-paper cross-layer links, and 62,602 venue-paper cross-layer links.

We evaluate the proposed algorithms in two scenarios with this data set, including (1) an author-paper two-layered network; and (2) a venue-paper two-layered network. For both scenarios, we choose the paper-paper citation network as the target layer. Fig. 9 presents the results on the author-paper two-layered network. We can see that three out of four OPERA algorithms outperform all the baseline methods in all the three cases. OPERA-PC does not perform as well as the remaining three OPERA instances due to the gap between the leading eigenvalue and the actual path capacity. However, the issue can be partially addressed with OPERA-PC-Up by introducing an update step. Among the baseline methods, the *backward propagation* strategy is better since the target layer is directly dependent on the control layer, which makes it possible to trace back the high-impact authors given the set of high-impact papers. The poor performance of the *forward propagation* methods implies that a socially active author does not necessarily have high-impact papers. The results on the venue-paper network is similar as shown in Fig. 10. Different from the author-paper network, the backward propagation strategies perform worse than the forward propagation strategies. This is probably due to the fact that not all the important (high-impact) papers appear in the important (high-impact) venues.

D4 - BIO. This data set contains three types of biological networks [15] including (1) a chemical similarity network with 6,026 chemicals, 69,109 links; (2) a gene similarity network with 25,394 genes, 154,167 links; and (3) a disease similarity network with 4,256 diseases, 30,551 links. The dependencies between those layers depict *which chemical-affects-which gene*, *which chemical-treats-which disease*, and *which gene-associates-which disease* relations, each of which contains 53,735, 19,771 and 1,950 dependency links respectively. The evaluation results are as shown in Fig. 11. Despite the fact that the proposed OPERA algorithms outperform all other baseline methods, there are two interesting observations that worth to be mentioned. First is that the impact of chemical nodes on disease networks become saturated at a small budge value for all connectivity measures, which implies that only

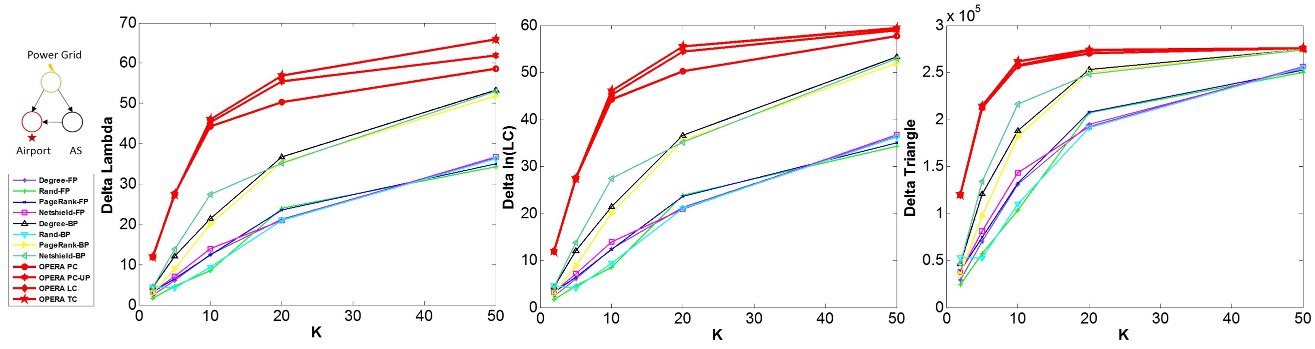


Fig. 7. Evaluations on the InfraNet data set, with a three-layered triangle-shaped dependency network. The connectivity change vs. budget. Larger is better. All the four instances of the proposed Opera algorithm (in red) outperform the baseline methods.

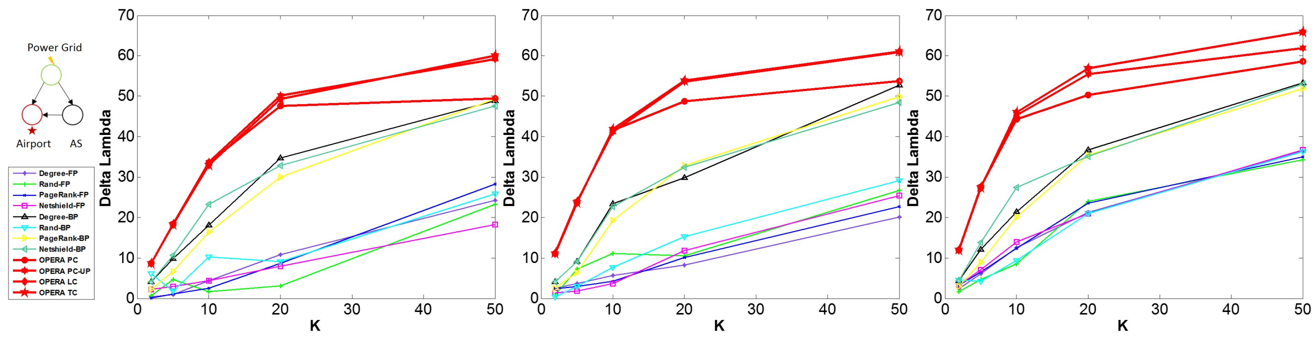


Fig. 8. $\Delta\lambda$ wrt K. Change the average number of dependents between Power Grid and AS from 5, 10 to 15 (left to right)

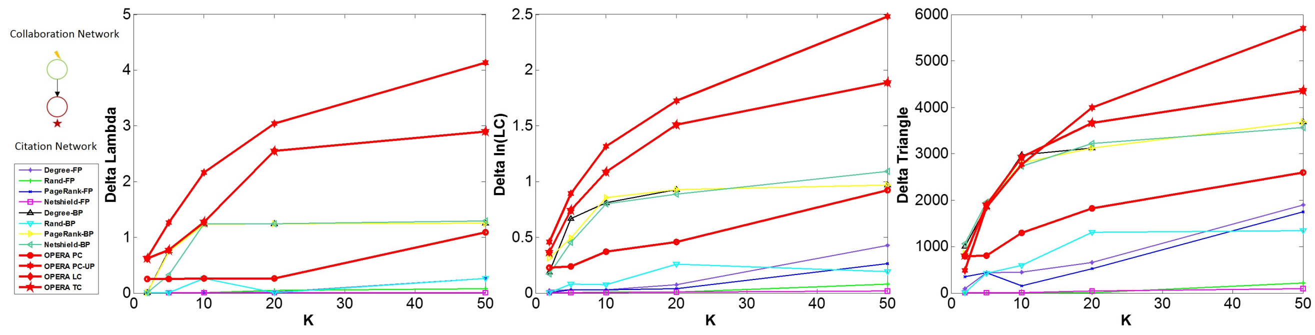


Fig. 9. Evaluations on the SocInNet data set, with a two-layered author-paper dependency network. The connectivity change vs. budget. Larger is better. Three out-of four proposed Opera algorithms (in red) outperform the baseline methods.

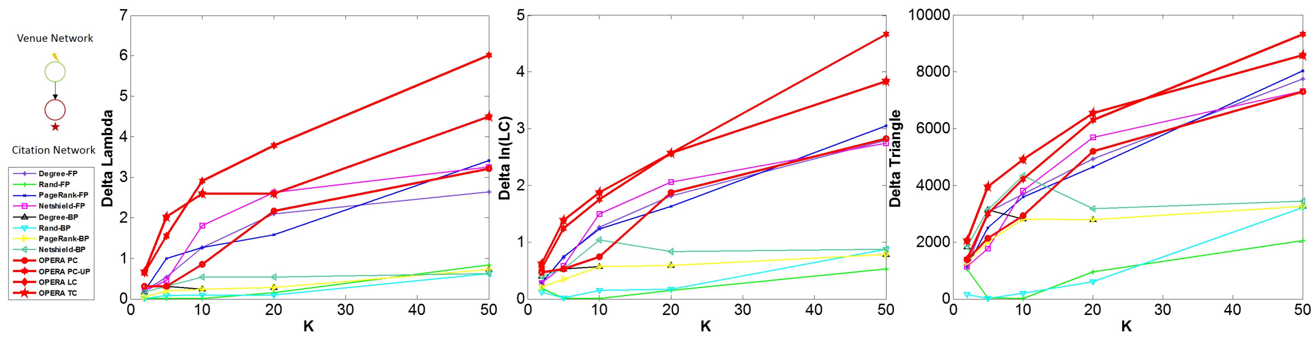


Fig. 10. Evaluations on the SocInNet data set, with a two-layered venue-paper dependency network. The connectivity change vs. budget. Larger is better. Three out-of four proposed Opera algorithms (in red) outperform the baseline methods.

a few chemicals are effective in treating most of the diseases in the given data set. Second, the ineffectiveness of *forward propagation* methods indicates that chemicals with various compounds (high within-layer centrality nodes) may have little effects in disease treatment.

5.3 Efficiency Results

Fig. 12 presents the scalability of the proposed Opera algorithms. We can see that all four instances of Opera scale linearly with respect to the size of the input multi-layered network (i.e., $N + M + L$), which is consistent with the complexity analysis in Subsection 4.3. The wall-clock time for Opera-PC-Up is the longest compared with the remaining three instances, due to the additional update step.

6 Related Work

In this section, we review the related work, which can be categorized into two groups: (a) network connectivity optimization, and

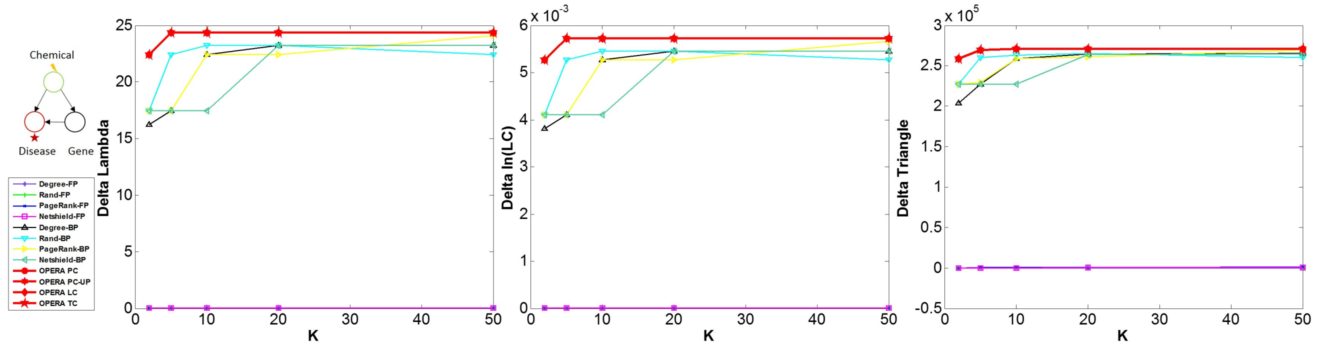
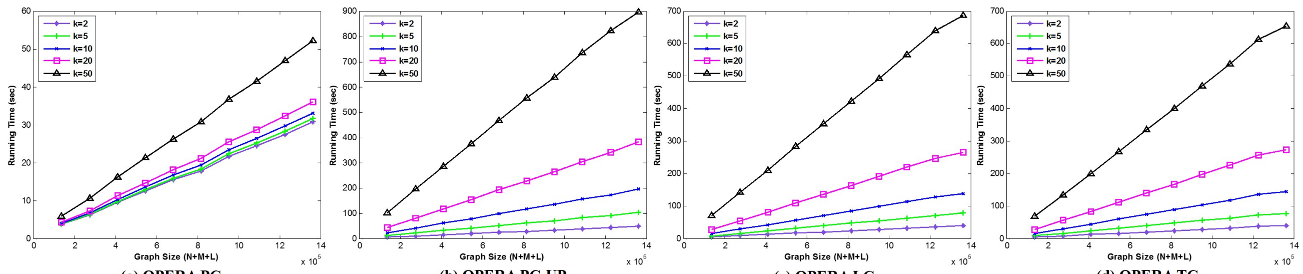


Fig. 11. Evaluations on the BIO data set, with a three-layered triangle-shaped dependency network. The connectivity change vs. budget. Larger is better. Three out-of four proposed OPERA algorithms (in red) outperform the baseline methods.



(a) OPERA PC (b) OPERA PC-UP (c) OPERA LC (d) OPERA TC

Fig. 12. Wall-clock time vs. the size of input networks. The proposed OPERA algorithms scale linearly wrt $(N + M + L)$.

(b) multi-layered network analysis.

Network Connectivity Optimization. Connectivity is a fundamental property of networks, and has been a core research theme in graph theory and mining for decades. Depending on the specific applications, many network connectivity measures have been proposed in the past. Examples include the size of giant connected component (GCC), graph diameter, the mixing time [19], the vulnerability measure [20], the epidemic thresholds [21], the natural connectivity [22] and the number of triangles in the network, each of which often has its own, different mathematical definitions. In [10], Milo et al. showed that network motifs are the simple building blocks of complex networks; and network motifs are essentially different patterns of subgraph structures. Partially inspired by this discovery, we find that many network connectivity measures can be expressed as the aggregation of the connectivity of its subgraph structures, which leads to a unified viewpoint of some prevalent network connectivity measures (Section 3).

From algorithm's perspective, network connectivity optimization aims to maximize or minimize the corresponding connectivity measures by manipulating the underlying topology (e.g., add/remove nodes/links). Earlier work, either explicitly or implicitly, assumes that nodes/links with higher centrality scores would have a greater impact on network connectivity. This assumption has led to many research efforts on finding good node/link centrality measures (or node/link importance measure in general) . Some widely used centrality measures include shortest path based centrality [23], PageRank [16], HITS [24], coreness score [25], local Fiedler vector centrality [26] and random walks based centrality [27]. Different from those node centrality oriented methods, some recent work aims to take one step further by *collectively* finding a subset of nodes/links with the highest impact on the network connectivity measure. For example, Tong et al. [8], [28], [29], [30] proposed both node-level and edge-level manipulation strategies to optimize the leading eigenvalue of the network, which is the key network connectivity measure behind a variety of cascading models. In [7], Chan et al. further generalized these strategies to manipulate the network robustness measure through the truncated loop capacity [22]. Another important aspect of network connectivity optimization problem lies in the network dynamics. Chen et al. proposed an efficient online algorithm to track some important network connectivity measures (e.g., the leading eigenvalue, the robustness measure) on a temporal dynamic network in [31], [32].

Multi-Layered Network Analysis. Multi-layered networks have been attracting a lot of research attention in recent years. Different models have been proposed to formulate the multi-layered network data structure. In [33], multi-layered networks are represented as a high-order tensor, which is coupled by a second-order within-layer networks tensor and a second-order cross-layer dependency tensor. While in [34], the corresponding data structure is represented as a quadruplet $M = \{V_M, E_M, V, \mathcal{L}\}$, in which each distinct nodes in V can appear in multiple elementary layers in $\mathcal{L} = \{L_1, \ldots, L_d\}$. Then, $V_M \subseteq V \times L_1 \times \ldots \times L_d$ represents the nodes in each layer, and $E_M = V_M \times V_M$ represents both within-layer and cross-layer links in the entire system. In [9], the model is simplified into a pair $M = (\mathcal{G}, \mathcal{C})$, where $\mathcal{G}$ gives all the within-layer networks and $\mathcal{C}$ provides all the cross-layer dependencies. Different from the above models, the formulation used in our paper gives more emphasis on the abstracted dependency network structure $\mathbf{G}$, which makes it easier to unravel the impact path for a set of nodes from a given layer. In [35], Kivela et al. presented a comprehensive survey on different types of multi-layered networks, which include multi-modal networks [36], multi-dimensional networks [37], multiplex networks [38] and interdependent networks [1]. The problem addressed in this paper is most related to the interdependent networks. In [3] and [39], the authors presented an in-depth introduction on the fundamental concepts of interdependent, multi-layered networks as well as the key research challenges. In a multi-layered network, the failure of a small number of the nodes might lead to catastrophic damages on the entire system as shown in [1] and [40]. In [1], [2], [4], [5], [6], different types of *two-layered* interdependent networks were thoroughly analyzed. In [39], Gao et al. analyzed the robustness of multi-layered networks with star- and loop-shaped dependency structures. Similar to the robustness measures in [41], most of the current works use the size of GCC (giant connected component) in the network as the evaluation standard [42], [43], [44]. Nonetheless, the fine-granulated connectivity details might not be captured by the GCC measure. To address the above issues, Chen et al. generalized a set of classic network connectivity measures into a unified framework and proposed an efficient algorithm to optimize the connectivity in more general structured multi-layered networks [45]. In [46], De Domenico et al. proposed a method to identify versatile nodes in multi-layered networks by evaluating their eigenvector centrality and pagerank centrality. The selected versatile nodes are fundamentally different from our high impact nodes in three aspects. First is that their centrality measures can not capture the collective impact of a node set on the network. Second is that our proposed network connectivity is directly related to only within-layer links, while cross-layer dependency is the trigger for connectivity changes. The two types of links should be treated differently rather than mixed up for a unified centrality calculation. Last, the globally crucial nodes in the entire system may not be able to provide an optimal solution to minimized the connectivity in specific target layer(s). On the other hand, existing works assume that the observed cross-layer dependencies in multi-layered networks are complete, which is not the case in real-world applications due to noise, limited accessibility, etc. In [47], a collaborative filtering based method is proposed to infer the missing cross-layer dependencies in multi-layered network. Other remotely related studies include cross-network ranking [48] and clustering [49], [50] in multi-layered networks, and multi-view data analysis [51], [52], [53].

7 Conclusion

In this paper, we study the connectivity optimization problem on multi-layered networks (OPERA). Our main contributions are as follows. First, we unify a family of prevalent network connectivity measures (SUBLINE). Second, we prove that for *any* network connectivity measures in the SUBLINE family, the connectivity optimization problem with the MULAN model enjoys the diminishing returns property, which naturally lends itself to a family of provable near-optimal algorithms with linear complexity. Finally, we conduct extensive empirical evaluations on real network data, which validate the effectiveness and efficiency of the proposed algorithms.

Acknowledgments

This material is supported by the National Science Foundation under Grant No. IIS1017415, by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, by Defense Advanced Research Projects Agency (DARPA) under Contract Number W911NF-11-C-0200 and W911NF-12-C-0028, by National Institutes of Health under the grant number R01LM011986, Region II University Transportation Center under the project number 49997-33 25.

The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic cascade of failures in interdependent networks," *Nature*, vol. 464, no. 7291, pp. 1025–1028, 2010.

[2] J. Gao, S. V. Buldyrev, H. E. Stanley, and S. Havlin, "Networks formed from interdependent networks," *Nature physics*, vol. 8, no. 1, pp. 40–48, 2012.

[3] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies," *Control Systems, IEEE*, vol. 21, no. 6, pp. 11–25, 2001.

[4] A. Sen, A. Mazumder, J. Banerjee, A. Das, and R. Compton, "Multi-layered network using a new model of interdependency," *arXiv preprint arXiv:1401.1783*, 2014.

[5] R. Parshani, S. V. Buldyrev, and S. Havlin, "Interdependent networks: Reducing the coupling strength leads to a change from a first to second order percolation transition," *Physical review letters*, vol. 105, no. 4, p. 048701, 2010.

[6] J. Shao, S. V. Buldyrev, S. Havlin, and H. E. Stanley, "Cascade of failures in coupled network systems with multiple support-dependent relations," *arXiv preprint arXiv:1011.0234*, 2010.

[7] H. Chan, L. Akoglu, and H. Tong, "Make it or break it: manipulating robustness in large networks," in *Proceedings of 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 325–333.

[8] H. Tong, B. A. Prakash, C. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau, "On the vulnerability of large graphs," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 1091–1096.

[9] S. Boccaletti, G. Bianconi, R. Criado, C. I. Del Genio, J. Gómez-Gardenes, M. Romance, I. Sendina-Nadal, Z. Wang, and M. Zanin, "The structure and dynamics of multilayer networks," *Physics Reports*, vol. 544, no. 1, pp. 1–122, 2014.

[10] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[11] C. E. Tsourakakis, "Fast counting of triangles in large real networks without counting: Algorithms and laws," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 608–617.

[12] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[13] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 420–429.

[14] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functionsi," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[15] A. P. Davis, C. J. Grondin, K. Lennon-Hopkins, C. Saraceni-Richards, D. Sciaky, B. L. King, T. C. Wiegers, and C. J. Mattingly, "The comparative toxicogenomics database's 10th year anniversary: update 2015," *Nucleic acids research*, vol. 43, no. D1, pp. D914–D920, 2015.

[16] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Tech. Rep., 1998, paper SIDL-WP-1999-0120 (version of 11/11/1999). [Online]. Available: http://dbpubs.stanford.edu/pub/1999-66

[17] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[18] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 990–998.

[19] M. Jerrum and A. Sinclair, "Conductance and the rapid mixing property for markov chains: the approximation of permanent resolved," in *Proceedings of the twentieth annual ACM symposium on Theory of computing*. ACM, 1988, pp. 235–244.

[20] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, pp. 378–382, 2000.

[21] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos, "Epidemic thresholds in real networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 4, p. 1, 2008.

[22] W. Jun, M. Barahona, T. Yue-Jin, and D. Hong-Zhong, "Natural connectivity of complex networks," *Chinese Physics Letters*, vol. 27, no. 7, p. 078902, 2010.

[23] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[24] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," in *ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[25] J. Moody and D. R. White, "Social cohesion and embeddedness: A hierarchical conception of social groups," *American Sociological Review*, pp. 1–25, 2003.

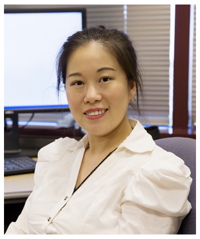
[26] P.-Y. Chen and A. O. Hero, "Local fiedler vector centrality for detection of deep and overlapping communities in networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1120–1124.

[27] M. E. Newman, "A measure of betweenness centrality based on random walks," *Social networks*, vol. 27, no. 1, pp. 39–54, 2005.

[28] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos, "Gelling, and melting, large graphs by edge manipulation," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 245–254.

[29] C. Chen, H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau, "Node immunization on large graphs: Theory and algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 113–126, 2016. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2015.2465378
[30] C. Chen, H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos, "Eigen-optimization on large graphs by edge manipulation," *TKDD*, vol. 10, no. 4, p. 49, 2016. [Online]. Available: http://doi.acm.org/10.1145/2903148
[31] C. Chen and H. Tong, "Fast eigen-functions tracking on dynamic graphs," in *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 2015.
[32] ——, "On the eigen-functions of dynamic graphs: Fast tracking and attribution algorithms," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, pp. n/a–n/a, 2016. [Online]. Available: http://dx.doi.org/10.1002/sam.11310
[33] M. De Domenico, A. Solé-Ribalta, E. Cozzo, M. Kivelä, Y. Moreno, M. A. Porter, S. Gómez, and A. Arenas, "Mathematical formulation of multilayer networks," *Physical Review X*, vol. 3, no. 4, p. 041022, 2013.
[34] R. J. Sánchez-García, E. Cozzo, and Y. Moreno, "Dimensionality reduction and spectral properties of multilayer networks," *Physical Review E*, vol. 89, no. 5, p. 052815, 2014.
[35] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *Journal of Complex Networks*, vol. 2, no. 3, pp. 203–271, 2014.
[36] L. S. Heath and A. A. Sioson, "Multimodal networks: Structure and operations," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 6, no. 2, pp. 321–332, 2009.
[37] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, and D. Pedreschi, "Foundations of multidimensional network analysis," in *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. IEEE, 2011, pp. 485–489.
[38] F. Battiston, V. Nicosia, and V. Latora, "Structural measures for multiplex networks," *Physical Review E*, vol. 89, no. 3, p. 032804, 2014.
[39] J. Gao, S. V. Buldyrev, S. Havlin, and H. E. Stanley, "Robustness of a network of networks," *Physical Review Letters*, vol. 107, no. 19, p. 195701, 2011.
[40] A. Vespignani, "Complex networks: The fragility of interdependency," *Nature*, vol. 464, no. 7291, pp. 984–985, 2010.
[41] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann, "Mitigation of malicious attacks on networks," *Proceedings of the National Academy of Sciences*, vol. 108, no. 10, pp. 3838–3841, 2011.
[42] M. Parandehgheibi and E. Modiano, "Robustness of interdependent networks: The case of communication networks and the power grid," in *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 2164–2169.
[43] D. T. Nguyen, Y. Shen, and M. T. Thai, "Detecting critical nodes in interdependent power networks for vulnerability assessment." *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 151–159, 2013.
[44] A. Bernstein, D. Bienstock, D. Hay, M. Uzunoglu, and G. Zussman, "Power grid vulnerability to geographically correlated failuresanalysis and control implications," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 2634–2642.
[45] C. Chen, J. He, N. Bliss, and H. Tong, "On the connectivity of multi-layered networks: Models, measures and optimal control," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 715–720.
[46] M. De Domenico, A. Solé-Ribalta, E. Omodei, S. Gómez, and A. Arenas, "Ranking in interconnected multilayer networks reveals versatile nodes," *Nature communications*, vol. 6, 2015.
[47] C. Chen, H. Tong, L. Xie, L. Ying, and Q. He, "FASCINATE: fast cross-layer dependency inference on multi-layered networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, pp. 765–774. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939784
[48] J. Ni, H. Tong, W. Fan, and X. Zhang, "Inside the atoms: ranking on a network of networks," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1356–1365.
[49] ——, "Flexible and robust multi-network clustering," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 835–844.
[50] R. Liu, W. Cheng, H. Tong, W. Wang, and X. Zhang, "Robust multi-network clustering via joint cross-domain cluster alignment," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 291–300.
[51] J. Li, X. Hu, L. Wu, and H. Liu, "Robust unsupervised feature selection on networked data," in *Proceedings of SIAM International Conference on Data Mining*. SIAM, 2016.
[52] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *arXiv preprint arXiv:1304.5634*, 2013.
[53] D. Zhou, J. He, K. S. Candan, and H. Davulcu, "Muvir: multi-view rare category detection."

Chen Chen Chen Chen is a PhD student in School of Computing, Informatics and Decision Systems Engineering at Arizona State University. She received her bachelor's degree and master's degree in computer science from Beihang University and New York University in 2011 and 2013, respectively. Her research interests include large scale data mining in graphs and real-world network analysis.



Jingrui He Jingrui He is an assistant professor in the School of Computing, Informatics and Decision Systems Engineering at Arizona State University. She received her PhD in Computer Science from Carnegie Mellon University. She joined ASU in 2014 and directs the Statistical Learning Lab (STAR Lab). Her research focuses on heterogeneous machine learning, rare category analysis, semi-supervised learning and active learning, with applications in healthcare, social network analysis, semiconductor manufacturing, etc. She is the recipient of the NSF CAREER Award in 2016, IBM Faculty Award in 2015 and 2014 respectively, and has published more than 60 refereed articles. She has served on the organizing committee/senior program committee of many conferences, including ICML, KDD, IJCAI, SDM, ICDM, etc. She is also the author of the book on Analysis of Rare Categories (Springer-Verlag, 2012).

Nadya Bliss Nadya T. Bliss is the Director of the Global Security Initiative (GSI) at Arizona State University. Before joining ASU in 2012,Nadya spent 10 yearsat MIT Lincoln Laboratory,most recently as the founding Group Leader of the Computing and Analytics Group. In 2011, Nadya was awarded the inaugural MIT Lincoln Laboratory Early Career Technical Achievement Award. Nadya received bachelor and master degrees in Computer Science from Cornell University, a PhD in Applied Mathematics for the Life and Social Sciences from Arizona State University, and is a Senior Member of IEEE.

Hanghang Tong Hanghang Tong is currently an assistant professor as School of Computing, Informatics and Decision Systems Engineering at Arizona State University. Before that, he was an assistant professor at Computer Science Department, City College, City University of New York, a research staff member at IBM T.J. Watson Research Center and a Post-doctoral fellow in Carnegie Mellon University. He received his M.Sc and Ph.D. degree from Carnegie Mellon University in 2008 and 2009, both majored in Machine Learning. His research interest is in large scale data mining for graphs and multimedia. He has received several awards, including best paper award in CIKM 2012, SDM 2008 and ICDM 2006. He has published over 80 referred articles and more than 20 patents. He has served as a program committee member in top data mining, databases and artificial intelligence venues.