Network Connectivity Optimization: Fundamental Limits and Effective Algorithms

Chen Chen
Arizona State University
chen_chen@asu.edu

Ruiyue Peng
Translational MRI, LLC
rpeng8@asu.edu

Lei Ying
Arizona State University
lei.ying.2@asu.edu

Hanghang Tong
Arizona State University
hanghang.tong@asu.edu

ABSTRACT

Network connectivity optimization, which aims to manipulate network connectivity by changing its underlying topology, is a fundamental task behind a wealth of high-impact data mining applications, ranging from immunization, critical infrastructure construction, social collaboration mining, bioinformatics analysis, to intelligent transportation system design. To tackle its exponential computation complexity, greedy algorithms have been extensively used for network connectivity optimization by exploiting its diminishing returns property. Despite the empirical success, two key challenges largely remain open. First, on the theoretic side, the *hardness*, as well as the *approximability* of the general network connectivity optimization problem are still nascent except for a few special instances. Second, on the algorithmic side, current algorithms are often hard to balance between the optimization quality and the computational efficiency. In this paper, we systematically address these two challenges for the network connectivity optimization problem. First, we reveal some fundamental limits by proving that, for a wide range of network connectivity optimization problems, (1) they are NP-hard and (2) $(1 - 1/e)$ is the optimal approximation ratio for any polynomial algorithms. Second, we propose an effective, scalable and general algorithm (CONTAIN) to carefully balance the optimization quality and the computational efficiency.

CCS CONCEPTS

• **Mathematics of computing → Paths and connectivity problems**;

ACM Reference Format:
Chen Chen, Ruiyue Peng, Lei Ying, and Hanghang Tong. 2018. Network Connectivity Optimization: Fundamental Limits and Effective Algorithms. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220019

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220019

1 INTRODUCTION

Network connectivity optimization is an essential task behind a myriad of high-impact data mining applications such as immunization, critical infrastructure construction, social collaboration mining, bioinformatics analysis, intelligent transportation system design, etc. In some of these applications, a less connected network might be preferred, which is termed as 'network destruction' in [19]. Specifically, it requires a network optimization algorithm to find a set of 'silver bullet' nodes/edges in the network to minimize its connectivity [19]. For example, in the immunization scenario, it is essential to identify crucial entities and links in the contagion network to effectively contain the spread of a disease. On the other hand, network connectivity optimization techniques may help fortify the robustness of the network. In infrastructure networks (e.g., power grids and transportation systems), the full functioning of the systems is strongly dependent on the connectivity of the underlying networks. Hence, it is of key importance for the maintenance team to identify critical facilities and transmission lines whose failure would sabotage the connectivity of the entire network, so that precaution and protection measures can be implemented proactively.

The main difficulty of the network connectivity optimization problem lies in its high computational complexity. To be specific, the fundamental limit for minimizing the network connectivity by removing a set of nodes/edges is rooted in its combinatorial nature. Given a budget k, the number of all possible node sets of size k is $\binom{n}{k}$, and the number of edge sets of size k is $\binom{m}{k}$, where n and m are the numbers of nodes and edges in the network. Such exponential complexity makes the brute-force algorithm computationally expensive and intractable, even in medium-sized networks.

To tackle the exponential computation complexity, state-of-the-art methods often resort to greedy algorithms. Taking network connectivity minimization problem by node deletions as an example, the greedy algorithm would iteratively delete the node with the highest *connectivity impact score* until the budget is reached. Specifically, the impact score of a node is defined as the marginal decrease of the network connectivity due to its removal from the intermediately optimized network in the previous iteration. Thanks to the diminishing returns property of the network connectivity optimization problem on a wide range of connectivity measures [7], the greedy algorithm guarantees a $(1 - 1/e)$ near-optimal approximation solution. A key step in such greedy algorithms is to estimate the impact score for each candidate node/edge. Some straightforward methods often involve an eigen decomposition operation, which would make the overall algorithm *polynomial* w.r.t. the input

network size, and thus do not scale to large networks with millions of nodes/edges. To address this issue, matrix perturbation based methods are often used to approximate the node/edge impact score by viewing the deletion of a node/edge as a perturbation to the current networks [9, 10]. Such approximation methods often exhibit empirical superiority over alternative methods, while ensuring a linear scalability.

Despite the empirical success of existing methods on some specific network connectivity optimization problems, two key challenges largely remain open. First, on the theoretic side, the *hardness* of the general connectivity optimization problem is unvalidated except for a few special instances (e.g., leading eigenvalue [10], triangle capacity [25]). Moreover, although the greedy algorithm guarantees a $(1-1/e)$ near-optimal solution, it remains unknown if such an approximation ratio is *optimal* over all the polynomial algorithms. Second, on the algorithmic side, the dilemma of the optimization quality vs. computational efficiency trade-off has largely remained. On one hand, although there exist tractable greedy algorithms for some special connectivity measures, they do not scale to large networks because of their super-linear complexity [25]. On the other hand, although matrix perturbation methods offer a linear time complexity, their optimization quality is largely dependent on the spectrum of the underlying network (e.g., the optimization quality would deteriorate quickly in small eigen-gap networks [8, 20]).

In this paper, we systematically address the above two challenges for the network connectivity optimization problem. The main contributions of the paper can be summarized as follows.

- *Revealing the Fundamental Limits.* We prove that for a wide range of connectivity optimization problems, (1) they are NP-hard and (2) $(1-1/e)$ is the best approximation ratio for any polynomial algorithms, unless $NP \subseteq DTIME(n^{O(\log\log n)})$[1].
- *Developing New Algorithms.* We propose an effective algorithm (CONTAIN) for network connectivity optimization. The centerpieces of the proposed method include (a) an effective impact score approximation method and (b) an efficient eigen-pair update method. The proposed CONTAIN algorithm bears three distinct advantages over the existing methods, including (1) *effectiveness*, being able to handle small eigen-gap networks, consistently outperforming the state-of-the-art methods over a diverse set of real networks; (2) *scalability*, with a linear complexity w.r.t. the network size; and (3) *generality*, applicable to a variety of different network connectivity measures (e.g., leading eigenvalue, triangle capacity and natural connectivity) as well as network operations (node vs. edge deletion).

2 PROBLEM DEFINITION

In this section, we formally introduce the network connectivity optimization problem and review the general strategy of greedy algorithms.

Table 1 gives the main symbols used throughout the paper. Following the convention, we use bold upper-case for matrices (e.g. $\mathbf{A}$), bold lower-case for vectors (e.g. $\mathbf{a}$) and calligraphic for sets (e.g. $\mathcal{A}$). We use $\tilde{}$ to denote the notations after node/edge deletion, and Δ to denote the perturbations (e.g. $\Delta\mathbf{A} = \tilde{\mathbf{A}} - \mathbf{A}$). $C(G)$ represents the network connectivity measure to be optimized in G; o indicates an element (a node/edge) in network G; $I(o)$ denotes the impact score of element o on $C(G)$; $\mathbf{\Lambda}$ and $\mathbf{U}$ denote the eigenvalue matrix and eigenvector matrix for the adjacency matrix $\mathbf{A}$ of the network.

Table 1: Main Symbols.

Symbol	Definition and Description
$G(V, E)$	an undirected network
$\mathbf{A}, \mathbf{B}$	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 the i^{th} row and the j^{th} column in $\mathbf{A}$
$\mathbf{a}(i)$	the i^{th} element of vector $\mathbf{a}$
$\mathbf{A}'$	transpose of matrix $\mathbf{A}$
$\Delta\mathbf{A}$	perturbation of $\mathbf{A}$
$\tilde{\mathbf{A}}$	the adjacency matrix after node/edge deletion on $\mathbf{A}$
m, n	number of edges and nodes in network G
$C(G)$	connectivity measure of network G
$F(\mathbf{\Lambda}^{(r)})$	associated eigen-function for $C(G)$
o	a network element in G (a node/edge)
$I(o)$	connectivity impact score of o on $C(G)$
$\lambda, \mathbf{u}$	the leading eigenvalue and eigenvector of $\mathbf{A}$ (in magnitude)
$\mathbf{\Lambda}, \mathbf{U}$	the eigenvalue and eigenvector matrix of $\mathbf{A}$
$\mathbf{\Lambda}^{(r)}, \mathbf{U}^{(r)}$	the top-r eigen-pairs of $\mathbf{A}$ (in magnitude)
k	the budget

2.1 Network Connectivity Measures

Many network connectivity measures can be defined as

$$C(G) = \sum_{\pi \in G} f(\pi) \tag{1}$$

where π is a subgraph of G, f is a non-negative function that maps any subgraph in G to a non-negative real number (i.e. $f : \pi \rightarrow \mathbb{R}^+$) [7]. Specifically, we have $f(\phi) = 0$ for empty set ϕ; when $f(\pi) > 0$, we call subgraph π as a *valid subgraph*. In other words, the network connectivity $C(G)$ can be viewed as a weighted aggregation of the connectivities of all valid subgraphs in the network.

By choosing an appropriate $f()$ function (please refer to [7] for details), Eq. (1) includes several prevalent network connectivity measures, e.g., path capacity (which is in close relation to the epidemic threshold), triangle capacity (which is rooted in social balance theory) and natural connectivity (which is closely related to network robustness). In terms of computation, it is often much more efficient to either approximate or compute these connectivity measures by the associated eigen-function $F(\mathbf{\Lambda}^{(r)})$, where $\mathbf{\Lambda}^{(r)}$ represents the top-r eigenvalues of $\mathbf{A}$. For example, the path capacity converges to the leading eigenvalue of the adjacency matrix of the network [4], the triangle capacity can be approximated by the sum of cubes of the eigenvalues [33], and the natural connectivity is calculated by the sum of exponentials of the eigenvalues [16].

2.2 Network Connectivity Optimization

With the network connectivity measure in Eq. (1), we formally define network connectivity optimization problem as follows.

Problem 1. **Network Connectivity Optimization (NETCOP)**
Given: *(1) a network G; (2) a connectivity mapping function $f : \pi \rightarrow \mathbb{R}^+$ which defines $C(G)$; (3) a type of network operation (node deletion vs. edge deletion) and (4) an integer budget k with $1 < k <$*

[1] $DTIME(t(n))$: the collection of languages that are decidable by $O(t(n))$ time deterministic Turing machine[31].

$\min\{|\mathcal{S}_\pi|, K\}$ *where* $\mathcal{S}_\pi = \{\pi | f(\pi) > 0\}$ *denotes the set of valid subgraphs and* K *denotes the number of valid network elements.*

Output: *a set of network elements* $\mathcal{X}$ *of size* k*, whose removal from* G *would minimize connectivity* $C(G)$*.*

It is worth noting that depending on the definition of $C(G)$, the valid subgraphs in $\mathcal{S}_\pi$ may have various structures. In the triangle minimization scenario, $\mathcal{S}_\pi$ contains all the triangles in the network. When the valid subgraph shares the same form as the operation type (i.e. a valid subgraph is a single node in node-level operation scenario, or a valid subgraph is an edge in edge-level operation scenario), we call this kind of valid subgraphs as *singletons*. In Problem 1, we also require that the budget $1 < k < \min\{|\mathcal{S}_\pi|, K\}$. This is a fairly generic constraint which can be easily met. For example, for the node deletion operation, the set of valid network elements is simply the entire node set of the input network (i.e., $K = n$); for a connected network with its connectivity measure $C(G)$ defined as the path capacity, we have that $|\mathcal{S}_\pi| > n$. Therefore, the above constraint simply means that we cannot delete all the nodes from the input network, which would make the problem trivial. On the other end of the spectrum, we require that the budget $k > 1$. Otherwise (with $k = 1$), the problem can be easily solved in polynomial time (e.g., by choosing the valid network element with the largest impact score). Problem 1 provides a general definition of the network connectivity optimization problem, which can be in turn instantiated into different instances, depending on (1) the specific choice of the connectivity measure $C(G)$ (or equivalently the choice of the $f()$ function), and (2) the type of network operation (node deletion vs. edge deletion). For example, in the robustness analysis of the power grid, we might choose the natural connectivity as $C(G)$ to evaluate the robustness of the system, and we are interested in identifying k most critical power transmission lines whose failure would cause a cascading failure of the entire grid. To abstract it as a network connectivity optimization problem, we have the input network set as the topological structure of the power grid; the connectivity to optimize as the natural connectivity; the operation type as edge deletion; and the valid network elements as all the edges (i.e., $K = m$ in this case).

2.3 Greedy Strategy for NETCOP

Due to the combinatorial nature of Problem 1, it is computationally infeasible to solve it in a brute-force manner. Thanks to the diminishing returns property of NETCOP, the greedy strategy has become a prevalent choice for solving Problem 1 with a guaranteed $(1 - 1/e)$ approximation ratio. For the ease of following discussions, we present the outline of such greedy strategy in Algorithm 1. In Algorithm 1, the solution set $\mathcal{X}$ is initialized with an empty set. At each iteration (step 2 to step 8), the element (a node or an edge) with the highest impact score is added to the solution set $\mathcal{X}$ until the budget is reached. The returned solution set $\mathcal{X}$ in step 9 guarantees a $(1 - 1/e)$ approximation ratio. For more details and proofs, please refer to [7].

3 FUNDAMENTAL LIMITS

In this section, we start with detailing the theoretic challenges of the network connectivity optimization (NETCOP) problem, and then reveal two fundamental limits, including its hardness and its approximability.

Algorithm 1 A Generic Greedy Strategy for NETCOP [7]

Input: (1) A network G; (2) a connectivity mapping function $f : \pi \to \mathbb{R}^+$ which defines $C(G)$; (3) a type of network operation and (4) a positive integer k

Output: a set of network elements $\mathcal{X}$ of size k.

1: initialize $\mathcal{X}$ to be empty
2: **for** $i = 1$ to k **do**
3: **for** each valid network element o in G **do**
4: calculate $I(o) \leftarrow C(G) - C(G \setminus \{o\})$
5: **end for**
6: add the element $\tilde{o} = \text{argmax}_o I(o)$ to $\mathcal{X}$
7: remove the element $\{\tilde{o}\}$ from network G
8: **end for**
9: return $\mathcal{X}$

3.1 Theoretic Challenges of NETCOP

The first theoretic challenge of NETCOP lies in its hardness. Since the NETCOP problem has various instances, intuitively, the hardness of those instances might vary dramatically from one to another. For example, if the elements in the valid subgraph set $\mathcal{S}_\pi$ are all singletons w.r.t. the corresponding operation type (i.e., $\mathcal{S}_\pi$ is the node set of the input network for the node-level optimization problem, or $\mathcal{S}_\pi$ is the edge set for the edge-level optimization problem), we can simply choose the top–k nodes/edges with the highest $f(\pi)$ scores, which immediately gives the optimal solution. However, if NETCOP is instantiated as an edge minimization problem under node deletion operations (i.e. the valid subgraph $\mathcal{S}_\pi$ consists of all the edges, the valid network element set is the entire node set) , the problem would become the (weighted) max-k vertex cover problem, which is known to be NP-hard. Such observations naturally give rise to the following question, *what is the key intrinsic property of valid subgraph set* $\mathcal{S}_\pi$ *in conjunction with the network operation type that determines whether or not the corresponding NETCOP instance is polynomially solvable?* To date, the hardness of the general NETCOP problem has largely remained unvalidated, except for a few special instances (see Section 6 for details). The second theoretic challenge of NETCOP lies in its approximability. The greedy algorithm outlined in Section 2 has a provable $(1 - 1/e)$ approximation ratio [7]. However, we still do not know if such an approximation ratio is *optimal*. In other words, it remains unknown if there exists any polynomial algorithm with an approximation ratio better than $(1 - 1/e)$ for NETCOP.

3.2 Fundamental Limit #1: NP-Hardness

We reveal the hardness result of the NETCOP problem in Theorem 1. It states that the NETCOP problem defined in Problem 1 are in general NP-hard, unless the valid subgraphs in set $\mathcal{S}_\pi$ are mutually independent to each other[2].

Theorem 1. **NP-Harness of NETCOP.** *The NETCOP problem with non-independent valid subgraphs in Problem 1 is NP-hard.*

Proof. See Appendix. □

[2]Two valid subgraphs are independent to each other if they do not have common valid network element.

3.3 Fundamental Limit #2: Approximability

Based on the hardness result of NETCOP, we further reveal the approximability of NETCOP in Theorem 2, which says that $(1-1/e)$ is indeed the best approximation ratio a polynomial algorithm can achieve unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$).

Theorem 2. **Approximability of NETCOP.** *$(1-1/e)$ is the best approximation ratio for the NETCOP problem in polynomial time, unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$).*

Proof. We prove this by contradiction. In the proof of Theorem 1 (see Appendix), we show that max k-hitting set problem is polynomially reducible to the NETCOP problem, which implies that if there is an α-approximation algorithm that can solve NETCOP in polynomial time with $\alpha > (1-1/e)$, there will be an α-approximation algorithm for max k-hitting set as well. However, it has been proved in [17] that the unit cost maximum k-coverage problem (an equivalent problem to max k-hitting set problem) can not be approximated with a factor better than $(1-1/e)$ unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$), which contradicts with our assumption. Hence, we conclude that there is no polynomial algorithm for the NETCOP problem with an approximation ratio greater than $(1-1/e)$, unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$). □

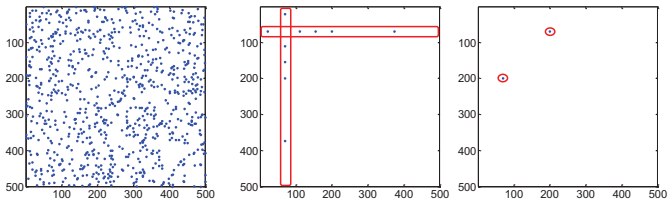
Since the greedy strategy in Algorithm 1 guarantees a $(1-1/e)$ approximation ratio, Theorem 2 implies that the greedy algorithm is the best polynomial algorithm for NETCOP in terms of its approximation ratio unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$).

4 ALGORITHM AND ANALYSIS

In this section, we start with detailing the algorithmic challenges of the network connectivity optimization (NETCOP) problem, and then present an effective algorithm, followed by some analysis in terms of its effectiveness and efficiency.

4.1 Algorithmic Challenges of NETCOP

In the greedy strategy (Algorithm 1), a key step is to calculate the impact score of each network element, i.e., $I(o) = C(G) - C(G \setminus \{o\})$ (Step 4). As we have mentioned in Section 2, the network connectivity measures $C(G)$ studied in this paper can be calculated or well approximated by a function of top-r eigenvalues of its adjacency matrix (i.e. $C(G) = F(\mathbf{\Lambda}^{(r)})$, where $F()$ is the function of eigenvalues). Therefore, the core step of calculating $I(o)$ is to compute $\mathbf{\Lambda}^{(r)}$ on $G \setminus \{o\}$, which takes $O(m)$ time (say using the classic Lanczos method). Consequently, simply recomputing $C(G \setminus \{o\})$ for each network element from scratch would make the entire algorithm $O(mn)$ for node-level optimization problems and $O(m^2)$ for edge-level optimization problems, neither of which is computationally feasible in large networks. To address this issue, existing literature often resorts to matrix perturbation theory. Its key idea is to view the deletion of a network element o as a perturbation to the original network (i.e. $\tilde{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A}$). Thus, the new eigenvalues (and hence the new connectivity measure $C(G \setminus \{o\})$) can be approximated from the eigenvalues and eigenvectors of the original network in constant time, making the overall algorithm linear w.r.t. the size of the input network [9, 10]. However, for networks with small eigen-gaps, the approximation accuracy of matrix perturbation theory based methods might deteriorate quickly, if not collapse at all.



(a) Random Perturbation (b) Node Deletion (c) Edge Deletion

Figure 1: Illustrations and comparison of random perturbation matrix (a), which is dense and potentially full-rank, vs. perturbation matrices by node deletion (b) and edge deletion (c), both of which are sparse and low-rank.

This issue might persist even if we switch to computationally more expensive high-order matrix perturbation theory [9, 10] . Thus, the main algorithmic challenge is how to accurately approximate the top-r eigenvalues of the input network after a node/edge deletion.

4.2 CONTAIN: The Proposed Algorithm

We propose a new updating algorithm for the top-r eigenvalues after node/edge deletion. In order to maintain the linear complexity of the entire greedy algorithm, we seek to update the top-r eigenvalues in *constant* time for each node/edge deletion operation.

Our key observation is as follows. In classic matrix perturbation theory (whether the first-order matrix perturbation theory or its high-order variants), a fundamental assumption is that the perturbation matrix $\Delta\mathbf{A}$ is a random matrix whose spectrum is well-bounded as illustrated in Figure 1(a). However, such assumption does not hold in the node/edge deletion scenario (Figure 1(b) and (c)), in which the perturbation matrix $\Delta\mathbf{A}$ is *sparse* and *low-rank*. Armed with this observation, we propose an effective eigen-pair update algorithm for node/edge deletion based on partial-QR decomposition. Unlike matrix perturbation based methods, which would inevitably introduce approximation error in the procedure, the proposed algorithm does not introduce any additional error when computing the impact score $I(o)$, and it runs in constant time for each node/edge operation.

The proposed CONTAIN algorithm is presented in Algorithm 2. Overall, it follows the greedy strategy (Algorithm 1). In detail, We first compute the top-r eigen-pairs of the network and compute the connectivity score of the original network (step 2-3). From step 4 to step 19, we iteratively select the element with the highest impact score. When evaluating the impact of each valid element, we first construct the perturbation matrix $\Delta\mathbf{A}$ for the corresponding element and then perform eigen decomposition on it (step 6-7). Particularly, for node deletion operation, suppose the removed node v has a set of neighbor nodes $\mathcal{N}_v$. Then the resulting perturbation matrix $\Delta\mathbf{A}$ has $\Delta\mathbf{A}(v, \mathcal{N}_v) = \Delta\mathbf{A}(\mathcal{N}_v, v) = -1$, which is a rank-2 sparse matrix. Therefore, $\mathbf{U}_\Delta$ and $\mathbf{\Lambda}_\Delta$ can be directly expressed as an $n\times 2$ matrix and a 2×2 matrix respectively. Moreover, let $n_v = |\mathcal{N}_v|$, the non-zero entries in the eigenvector matrix of $\Delta\mathbf{A}$ are

$$\mathbf{U}_\Delta(v,1) = \frac{1}{\sqrt{2}}, \mathbf{U}_\Delta(v,2) = \frac{1}{\sqrt{2}}$$
$$\mathbf{U}_\Delta(\mathcal{N}_v,1) = -\frac{1}{\sqrt{2n_v}}, \mathbf{U}_\Delta(\mathcal{N}_v,2) = \frac{1}{\sqrt{2n_v}} \quad (2)$$

Algorithm 2 The CONTAIN Algorithm

Input: (1) The adjacency matrix of the network $\mathbf{A}$; (2) the associated eigen-function $F()$ for connectivity $C(G)$; (3) rank r; (4) the network operation (node vs. edge deletion); and (5) a positive integer k.
Output: a set of network elements $\mathcal{X}$ of size k.

1: initialize $\mathcal{X}$ to be empty
2: compute $[\mathbf{U}^{(r)}, \mathbf{\Lambda}^{(r)}] \leftarrow$top-$r$ eigen-pairs of matrix $\mathbf{A}$
3: compute $C(G) \leftarrow F(\mathbf{\Lambda}^{(r)})$
4: **for** $i = 1$ to k **do**
5: **for** each valid element o in G **do**
6: $\Delta\mathbf{A} \leftarrow$ the perturbation matrix by element o's deletion
7: $[\mathbf{U}_\Delta, \mathbf{\Lambda}_\Delta] \leftarrow$eigen-pairs of $\Delta\mathbf{A}$
8: $\mathbf{R} \leftarrow$upper triangular matrix from $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta]$'s partial-QR decomposition
9: $\mathbf{\Lambda}_z \leftarrow$ eigenvalues of $\mathbf{Z} = \mathbf{R}[\mathbf{\Lambda}^{(r)}, \mathbf{0}; \mathbf{0}, \mathbf{\Lambda}_\Delta]\mathbf{R}'$
10: compute $I(o) \leftarrow C(G) - F(\mathbf{\Lambda}_z)$
11: **end for**
12: add $\tilde{o} = \text{argmax}_o I(o)$ to $\mathcal{X}$
13: update $C(G) \leftarrow C(G) - I(\tilde{o})$ and set $I(\tilde{o}) \leftarrow -1$
14: $\Delta\mathbf{A} \leftarrow$ the perturbation matrix by element $\tilde{o}$'s deletion
15: $[\mathbf{U}_\Delta, \mathbf{\Lambda}_\Delta] \leftarrow$ eigen-pairs of $\Delta\mathbf{A}$
16: $[\mathbf{Q}, \mathbf{R}] \leftarrow$ partial-QR decomposition of $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta]$
17: $[\mathbf{U}_z, \mathbf{\Lambda}_z] \leftarrow$ eigen-pairs of $\mathbf{Z} = \mathbf{R}[\mathbf{\Lambda}^{(r)}, \mathbf{0}; \mathbf{0}, \mathbf{\Lambda}_\Delta]\mathbf{R}'$
18: update $\mathbf{U}^{(r)} \leftarrow (\mathbf{Q}\mathbf{U}_z)^{(r)}$, $\mathbf{\Lambda}^{(r)} \leftarrow \mathbf{\Lambda}_z^{(r)}$, $\mathbf{A} \leftarrow \mathbf{A} + \Delta\mathbf{A}$
19: **end for**
20: return $\mathcal{X}$

and the eigenvalue matrix of $\Delta\mathbf{A}$ is

$$\mathbf{\Lambda}_\Delta = \begin{bmatrix} \sqrt{n_v} & 0 \\ 0 & -\sqrt{n_v} \end{bmatrix} \tag{3}$$

In the edge deletion scenario, the perturbation matrix $\Delta\mathbf{A}$ corresponding to the removal of edge $\langle u, v\rangle$ has only two non-zero entries $\Delta\mathbf{A}(u, v) = \Delta\mathbf{A}(v, u) = -1$ and $u \neq v$, which is also a rank-2 matrix. Then, the only non-zero entries in $\mathbf{U}_\Delta$ are

$$\begin{aligned} \mathbf{U}_\Delta(u, 1) = \frac{1}{\sqrt{2}}, \mathbf{U}_\Delta(u, 2) = \frac{1}{\sqrt{2}} \\ \mathbf{U}_\Delta(v, 1) = -\frac{1}{\sqrt{2}}, \mathbf{U}_\Delta(v, 2) = \frac{1}{\sqrt{2}} \end{aligned} \tag{4}$$

And the eigenvalue matrix $\mathbf{\Lambda}_\Delta$ is

$$\mathbf{\Lambda}_\Delta = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{5}$$

With the eigenvector matrix of $\Delta\mathbf{A}$, we proceed to perform partial-QR decomposition on $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta]$ in step 8. As $\mathbf{U}^{(r)}$ is already orthonormal, the $\mathbf{Q}$ matrix in the decomposition can be written as the concatenation of $\mathbf{U}^{(r)}$ and two orthogonal vectors in unit length as follows

$$\mathbf{Q} = [\mathbf{U}^{(r)}, \frac{\mathbf{q}_1}{\|\mathbf{q}_1\|}, \frac{\mathbf{q}_2}{\|\mathbf{q}_2\|}] \tag{6}$$

By the Gram-Schmidt process, we have

$$\begin{aligned} \mathbf{q}_1 &= \mathbf{U}_\Delta(:, 1) - \mathbf{U}^{(r)}\mathbf{r}_1 \\ \mathbf{q}_2 &= \mathbf{U}_\Delta(:, 2) - \mathbf{U}^{(r)}\mathbf{r}_2 + \mathbf{r}_1'\mathbf{r}_2 \frac{\mathbf{q}_1}{\|\mathbf{q}_1\|^2} \end{aligned} \tag{7}$$

where $\mathbf{r}_1 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:, 1)$ and $\mathbf{r}_2 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:, 2)$.
For node-level operations, we have

$$\begin{aligned} \mathbf{r}_1 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:, 1) = \frac{1}{\sqrt{2}}(\mathbf{U}^{(r)}(v, :) - \frac{1}{\sqrt{n_v}} \sum_{u \in \mathcal{N}_v} \mathbf{U}^{(r)}(u, :))' \\ \mathbf{r}_2 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:, 2) = \frac{1}{\sqrt{2}}(\mathbf{U}^{(r)}(v, :) + \frac{1}{\sqrt{n_v}} \sum_{u \in \mathcal{N}_v} \mathbf{U}^{(r)}(u, :))' \end{aligned} \tag{8}$$

While for edge-level operations, we have

$$\begin{aligned} \mathbf{r}_1 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:, 1) = \frac{1}{\sqrt{2}}(\mathbf{U}^{(r)}(u, :) - \mathbf{U}^{(r)}(v, :))' \\ \mathbf{r}_2 = \mathbf{U}^{(r)\prime}\mathbf{U}_\Delta(:, 2) = \frac{1}{\sqrt{2}}(\mathbf{U}^{(r)}(u, :) + \mathbf{U}^{(r)}(v, :))' \end{aligned} \tag{9}$$

Correspondingly, the upper-triangular matrix $\mathbf{R}$ can be written as

$$\mathbf{R} = \begin{bmatrix} \mathbf{I} & \mathbf{r}_1 & \mathbf{r}_2 \\ 0 & \|\mathbf{q}_1\| & -\frac{\mathbf{r}_1'\mathbf{r}_2}{\|\mathbf{q}_1\|} \\ 0 & 0 & \|\mathbf{q}_2\| \end{bmatrix} \tag{10}$$

By the definition of $\mathbf{q}_1, \mathbf{q}_2$ in Eq. (7) together with the orthonormal property of the eigenvectors, the norms of $\mathbf{q}_1$ and $\mathbf{q}_2$ can be computed indirectly with two $r \times 1$ vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ as

$$\begin{aligned} \|\mathbf{q}_1\| &= \sqrt{1 - \|\mathbf{r}_1\|^2} \\ \|\mathbf{q}_2\| &= \sqrt{1 - \|\mathbf{r}_2\|^2 - \frac{(\mathbf{r}_1'\mathbf{r}_2)^2}{1 - \|\mathbf{r}_1\|^2}} \end{aligned} \tag{11}$$

This enables us to compute $\|\mathbf{q}_1\|$ and $\|\mathbf{q}_2\|$ without explicitly constructing $\mathbf{q}_1$ and $\mathbf{q}_2$, which reduces the cost of step 8 from $O(nr)$ to $O(r)$. It can be proved that by setting $\mathbf{Z} = \mathbf{R}[\mathbf{\Lambda}^{(r)}, \mathbf{0}; \mathbf{0}, \mathbf{\Lambda}_\Delta]\mathbf{R}'$, the eigenvalues of $\mathbf{Z}$ are just the top eigenvalues of the perturbed matrix $\mathbf{A} + \Delta\mathbf{A}$, and the top eigenvectors of $\mathbf{A} + \Delta\mathbf{A}$ can be calculated by $\mathbf{Q}\mathbf{U}_z$ (step 18). Therefore, we only need $\mathbf{\Lambda}_z$ to compute the impact score of element o (step 10). After scanning all the valid elements in the current network, we choose the one with the largest impact score and add it to the element set $\mathcal{X}$ (step 12-13). Then, we update the network and its eigen-pairs (step 14-18). The procedure to update eign-pairs is similar to that of computing the impact score for a given network element (step 6-9), with the following subtle difference. In order to just compute the impact score of a given network element, we only need the updated eigenvalues. This is crucial as it saves the computation of (1) constructing $\mathbf{q}_1$ and $\mathbf{q}_2$, (2) finding the eigenvectors of $\mathbf{Z}$, and (3) updating the eigenvectors of perturbed matrix $\mathbf{A} + \Delta\mathbf{A}$, which in turn helps maintain constant time complexity for each inner for-loop (step 5-11).

4.3 Proof and Analysis

In this subsection, we analyze the proposed CONTAIN algorithm w.r.t. its effectiveness and efficiency.

4.3.1 Effectiveness. The effectiveness of CONTAIN is summarized in Lemma 3, which says that the computation of the impact score for each valid network element in the inner for-loop does not introduce any extra approximation error.

Lemma 3. **Effectiveness of CONTAIN.** *Suppose* $\mathbf{A}$ *is approximated with its top-r eigen-pairs with error* $\mathbf{E}$ *(i.e.* $\mathbf{A} = \mathbf{U}^{(r)}\mathbf{\Lambda}^{(r)}\mathbf{U}^{(r)\prime} + \mathbf{E}$*), then the* $\mathbf{\Lambda}_z$ *and* $\mathbf{QU}_z$ *returned in Algorithm 2 can be used to approximate* $\tilde{\mathbf{A}}$ *as its top eigen-pairs with no extra error.*

Proof. As $\mathbf{A} = \mathbf{U}^{(r)}\mathbf{\Lambda}^{(r)}\mathbf{U}^{(r)\prime} + \mathbf{E}$ and $\Delta\mathbf{A} = \mathbf{U}_\Delta\mathbf{\Lambda}_\Delta\mathbf{U}'_\Delta$, then $\tilde{\mathbf{A}}$ can be expressed as

$$\begin{aligned}\tilde{\mathbf{A}} &= \mathbf{U}^{(r)}\mathbf{\Lambda}^{(r)}\mathbf{U}^{(r)\prime} + \mathbf{U}_\Delta\mathbf{\Lambda}_\Delta\mathbf{U}'_\Delta + \mathbf{E} \\ &= [\mathbf{U}^{(r)}, \mathbf{U}_\Delta]\begin{bmatrix}\mathbf{\Lambda}^{(r)} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_\Delta\end{bmatrix}[\mathbf{U}^{(r)}, \mathbf{U}_\Delta]' + \mathbf{E} \end{aligned} \tag{12}$$

Perform partial-QR decomposition on $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta]$ as $[\mathbf{U}^{(r)}, \mathbf{U}_\Delta] = \mathbf{QR}$, we get orthonormal basis for $\tilde{\mathbf{A}}$ and an upper triangular matrix $\mathbf{R}$. Then the perturbed matrix $\tilde{\mathbf{A}}$ can be rewritten as

$$\tilde{\mathbf{A}} = \mathbf{QR}\begin{bmatrix}\mathbf{\Lambda}^{(r)} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_\Delta\end{bmatrix}\mathbf{R}'\mathbf{Q}' + \mathbf{E} \tag{13}$$

Let $\mathbf{Z} = \mathbf{R}[\mathbf{\Lambda}^{(r)}, \mathbf{0}; \mathbf{0}, \mathbf{\Lambda}_\Delta]\mathbf{R}'$ and perform eigen decomposition on $\mathbf{Z}$ as $\mathbf{Z} = \mathbf{U}_z\mathbf{\Lambda}_z\mathbf{U}'_z$, $\tilde{\mathbf{A}}$ is now equivalent to

$$\tilde{\mathbf{A}} = \mathbf{QU}_z\mathbf{\Lambda}_z\mathbf{U}'_z\mathbf{Q}' + \mathbf{E} = (\mathbf{QU}_z)\mathbf{\Lambda}_z(\mathbf{QU}_z)' + \mathbf{E} \tag{14}$$

Since both $\mathbf{Q}$ and $\mathbf{U}_z$ are orthonormal, we have $(\mathbf{QU}_z)(\mathbf{QU}_z)' = \mathbf{I}$. Thus, $\mathbf{\Lambda}_z$ and $\mathbf{QU}_z$ can be viewed as the top eigen-pairs of $\tilde{\mathbf{A}}$. As the approximation error remains to be $\mathbf{E}$ in Eq. (14), it implies that no extra error is introduced in the procedure, which completes the proof. □

4.3.2 Efficiency. The complexity of the proposed CONTAIN algorithm is summarized in Lemma 4, which says it is linear in both time and space.

Lemma 4. **Complexity of CONTAIN.** *The time complexity of CONTAIN for node-level connectivity optimization is* $O(k(mr + nr^3))$. *The time complexity of CONTAIN for edge-level connectivity optimization is* $O(k(mr^3 + nr^2))$. *The space complexity of CONTAIN is* $O(nr + m)$.

Proof. In the CONTAIN algorithm, computing top-r eigen-pairs and connectivity $C(G)$ would takes $O(nr^2 + mr)$ and $O(r)$ respectively. To compute the impact score for each node/edge (step 5-11), it takes $O(d_v r)$ (d_v is the degree of node v) for node v, and $O(r)$ for each edge to get the upper triangular matrix $\mathbf{R}$ in step 8. Since performing eigen-decomposition on $\mathbf{Z}$ at step 9 takes $O(r^3)$, the complexity to collect impact scores for all the nodes/edges are $O(nr^3 + mr)$ and $O(mr^3)$ respectively. Picking out the node/edge with highest impact score in current iteration would cost $O(n)$ for node level operations and $O(m)$ for edge level operations. At the end of the iteration, updating the eigen-pairs of the network takes the complexity of $O(nr^2 + r^3)$. As we have $r \ll n$, the overall time complexity to select k nodes would be $O(k(mr + nr^3))$; and the complexity to select k edges would be $O(k(mr^3 + nr^2))$

For space complexity, it takes $O(n+m)$ to store the entire network, $O(nr)$ to calculate and store the top-r eigen-pair of the network, $O(n)$ to store the impact scores for all the nodes in node level optimization scenarios and $O(m)$ to store the impact scores for all the edges, the eigen-pair update requires a space of $O(nr)$. Therefore, the overall space complexity for CONTAIN is $O(nr + m)$. □

5 EVALUATIONS

In this section, we evaluate the proposed CONTAIN algorithm. All experiments are designed to answer the following two questions:

- **Effectiveness.** How effective is the proposed CONTAIN algorithm in minimizing various connectivity measures?
- **Efficiency.** How efficient and scalable is the proposed CONTAIN algorithm?

5.1 Experiment Setup

5.1.1 Datasets. We perform experiments on 10 different datasets from 4 different domains, including Airport: an air traffic network that represents the direct flight connections between internal US airports[3]; Oregon: an autonomous system network which depicts the information transferring relationship between routers from [21]; Chemical: a network based on [12] that shows the similarity between different chemicals; Disease: a network that depicts the similarity between different diseases [12]; Gene: a protein-protein interaction network based on [12]; Astrph: a collaboration network between authors whose papers were submitted to Astro Physics category on Arxiv [22]; Hepth: a collaboration network between authors whose papers were submitted to High Energy Physics (Theory category) on Arxiv [21]; Aminer: a collaboration network between researchers in the Aminer datasets [32]; Eucore: the email correspondence network from a large European research institution [22]; and Fb: a social circle network collected from Facebook [26]. The statistics of those datasets are listed in Table 2.

Table 2: Statistics of Datasets.

Domain	Dataset	#Nodes	#Edges	Avg Degree
Infrastructure	Airport	2,833	7,602	5.37
	Oregon	5,296	10,097	3.81
Biology	Chemical	6,026	69,109	22.94
	Disease	4,256	30,551	14.36
	Gene	7,604	14,071	3.7
Collaboration	Astrph	18,772	198,050	21.1
	Hepth	9,877	25,985	5.26
	Aminer	1,211,749	4,756,194	7.85
Social	Eucore	1,005	16,064	31.97
	Fb	4,039	88,234	43.69

5.1.2 Comparing Methods. We compare the proposed algorithm with the following methods. (1) *Degree*: selecting top–k nodes (edges) with the largest degrees; specifically, for edge $\langle u, v\rangle$, let d_u and d_v denote the degrees for its endpoints respectively, the score for $\langle u, v\rangle$ is $\min\{d_u, d_v\}$[4]. (2) *PageRank*: selecting top–k nodes (edges) with the largest PageRank scores [29] (the corresponding edge score is the minimum PageRank score among its two endpoints); (3) *Eigenvector*: selecting top–k nodes (edges) with the largest eigenvector centrality scores [28] (the corresponding edge score is the minimum eigenvector centrality score among its endpoints); (4) *Netshield*/*Netmelt*: selecting top–k nodes (edges) that minimize the leading eigenvalue of the network [9, 10]; (5) *MIOBI*: a greedy algorithm that employs first-order matrix perturbation

[3] http://www.levmuchnik.net/Content/Networks/NetworkData.html.
[4] We use $\min\{d_u, d_v\}$ as edge score to ensure that both ends of the top ranked edges are high degree nodes.

method to estimate element impact score and update eigen-pairs [5]; (6) *MIOBI-S*: a variant of MIOBI that selects top–k nodes (edges) in one batch without updating the eigen-pairs of the network; (7) *MIOBI-H*: a variant of MIOBI that employs high order matrix perturbation method to update eigen-pairs [8]; (8) *Exact*: a greedy algorithm that recomputes the top-r eigen-pairs to estimate the impact score for each candidate node/edge. For the results reported in this paper, we set rank $r = 80$ for all the top–r eigen-pairs based approximation methods (methods (5)-(8) and the proposed CONTAIN method).

5.1.3 Evaluation Metrics. The performance of the algorithm is evaluated by the impact of its selected elements $I(\mathcal{X}) = C(G) - C(G \setminus \mathcal{X})$. The larger the $I(\mathcal{X})$ is, the more effective the algorithm is. For a given dataset, connectivity measure and network operation, we normalize $I(\mathcal{X})$ by that of the best method, so that the results across different datasets are comparable in the same plot.

5.1.4 Machine and Repeatability. All the experiments in the paper are performed on a machine with 2 processors (Intel Xeon 3.5GHz) with 256GB of RAM. The algorithms are programmed with MATLAB using a single thread. The code and the non-proprietary datasets will be released after the paper is published.

5.2 Effectiveness

5.2.1 Effectiveness of CONTAIN. We compare the proposed algorithm and the baseline methods on three connectivity measures (leading eigenvalue, number of triangles, and natural connectivity) by both node-level operations and edge-level operations on all datasets in our experiment. Since the *Exact* method needs to recompute the top-r eigen-pairs for each candidate node/edge which is very time-consuming, its results would be absent on some large datasets (e.g., AMINIER and ASTRPH) where it does not finish the computation within 24 hours. In our experiment, the budget for node-level operations is $k = 20$, the budget for edge-level operations is $k = 200$. The results are shown from Figure 2 to Figure 7. We can see that the proposed CONTAIN (the rightmost bar) (1) is very close to the *Exact* method (the black, hollow bar); and (2) consistently outperforms all the other alternative methods. In the meanwhile, the proposed CONTAIN algorithm is much faster than *Exact*, as will shown in the next subsection.

5.2.2 Effect of Rank r. The main parameter that affects the performance of CONTAIN is the rank r. To study the effect of r, we change r from 5 to 80 to minimize the number of triangles on the CHEMICAL dataset and compare them with the *Exact* method. The results are shown in Figure 8. From Figure 8, it is obvious to see that as r increases, the performance of CONTAIN increases accordingly, which is consistent with our effectiveness analysis. With $r = 80$, the performance of CONTAIN is very close to the *Exact* method with different k.

5.3 Efficiency

5.3.1 Efficiency of CONTAIN. Figure 9 presents the quality vs. running time trade-off of different methods for optimizing the natural connectivity (the most complicated connectivity measure) on the CHEMICAL dataset. In both node-level and edge-level optimization scenarios, the proposed CONTAIN achieves a very similar performance as *Exact*. In terms of the running time, CONTAIN is orders of magnitude faster than *Exact*. Although the running time of other baseline methods is similar to CONTAIN, their performance (y-axis) is not as good as CONTAIN.

5.3.2 Scalability of CONTAIN. The scalability results of CONTAIN are presented in Figure 10. As we can see, the proposed CONTAIN algorithm scales linearly w.r.t. the size of the input network (i.e. both the number of nodes and edges), which is consistent with Lemma 4.

6 RELATED WORK

In this section, we review the related literature from the following two perspectives, including (a) connectivity measures and (b) network connectivity optimization algorithms.

Connectivity Measures. At the macro-level, network connectivity can be viewed as a measure to evaluate how well the nodes are connected together. Examples include the size of giant connected component [3], graph diameter [1], the mixing time [14], the vulnerability measure [2], and the clustering coefficient [34]. At the micro-view level, network connectivity measures the capacity of edges, paths, loops, some complex motifs [27] or even the centrality of the nodes. Examples include the epidemic threshold [4], the natural connectivity (i.e., the robustness) [16], degree centrality [13], etc. Such connectivity measures have been extensively used for propagation analysis [30], graph clustering [35, 36], etc.

Connectivity Optimization Algorithms. State-of-the-art algorithms for network connectivity optimization are almost exclusively designed for a specific connectivity measure and/or with a specific network operation (node deletion vs. edge deletion). To name a few, Chen et al. have proposed both node-level and edge-level manipulation strategies to optimize leading eigenvalue and proved that the corresponding node-level optimization problem is NP-hard in [10] and [9], respectively. On the other hand, Le et al. have proposed an algorithm to minimize the leading eigenvalue for networks with small eigen-gaps in [20]. In [25], Li et al. show that both node-level and edge-level triangle minimization problem is NP-hard and have proposed several heuristic strategies for the triangle minimization problem. In [5] and [6], Chan et al. study the optimization problem for network robustness without analyzing its hardness. In order to effectively and efficiently compute the impact scores of network elements, the proposed CONTAIN algorithm resorts to partial-QR decomposition, which has been successfully used in several dynamic network mining tasks [11, 15, 23, 24].

7 CONCLUSIONS

In this paper, we study the network connectivity optimization problem by addressing two open challenges. On the theoretic side, we prove that a wide range of network connectivity optimization (NETCOP) problems are NP-hard and $(1 - 1/e)$ is the best approximation ratio that a polynomial algorithm can achieve for NETCOP problems unless $NP \subseteq DTIME(n^{O(\log \log n)})$. On the algorithmic aspect, we propose an effective, scalable and generalizable algorithm CONTAIN. Extensive experimental evaluations on a variety of real networks demonstrate that the proposed algorithm (1) consistently outperforms alternative methods, and (2) scales linearly w.r.t. the network size.

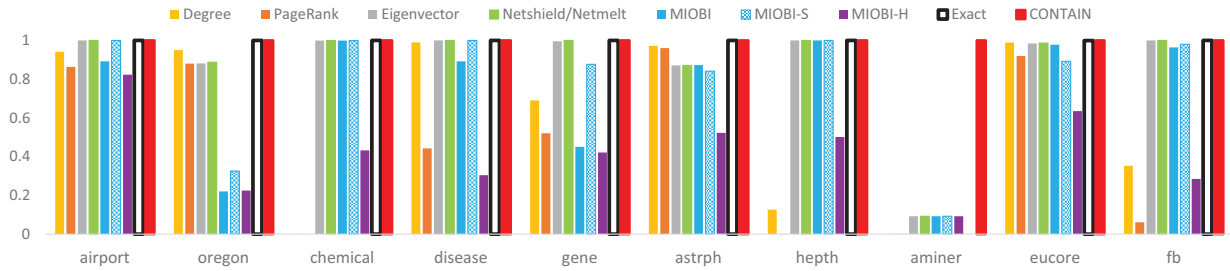


Figure 2: The optimization results on leading eigenvalue with node-level operations.

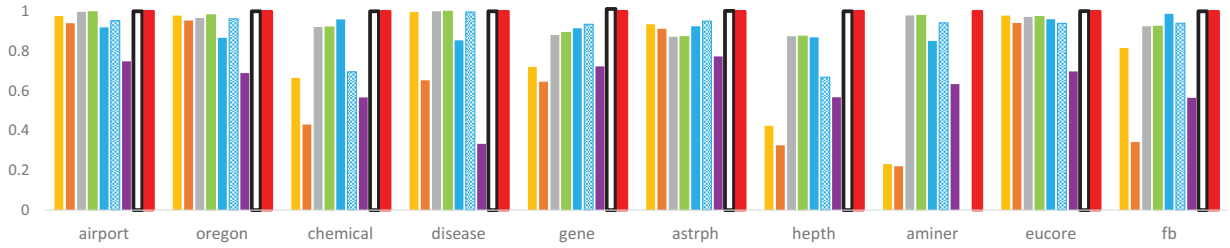


Figure 3: The optimization results on the number of triangles with node-level operations.

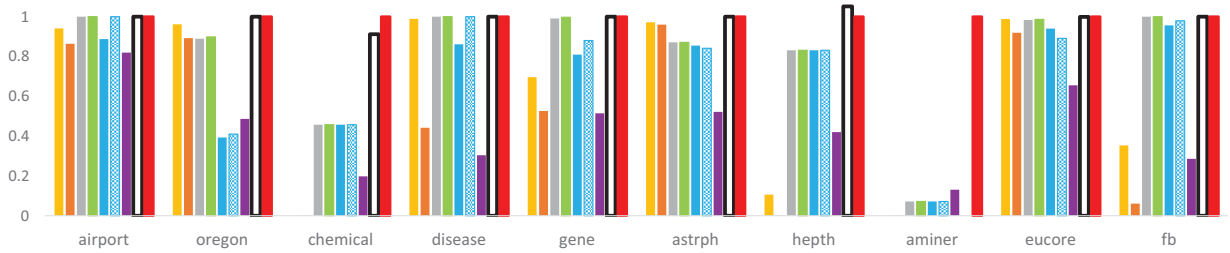


Figure 4: The optimization results on natural connectivity with node-level operations.

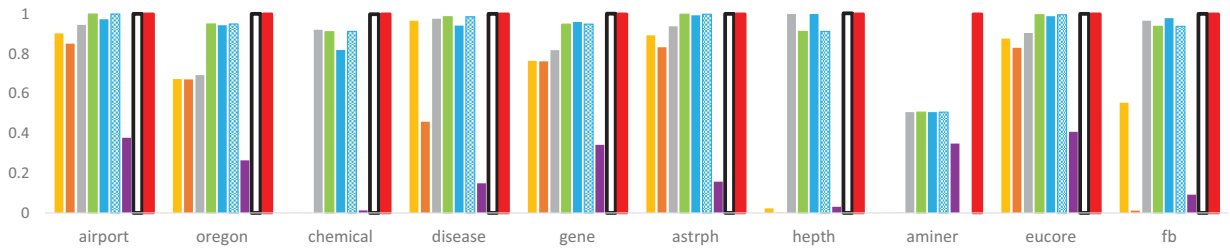


Figure 5: The optimization results on leading eigenvalue with edge-level operations.

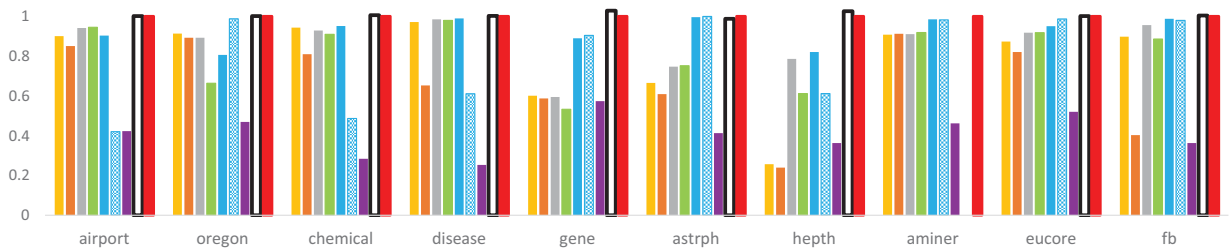


Figure 6: The optimization results on the number of triangles with edge-level operations.

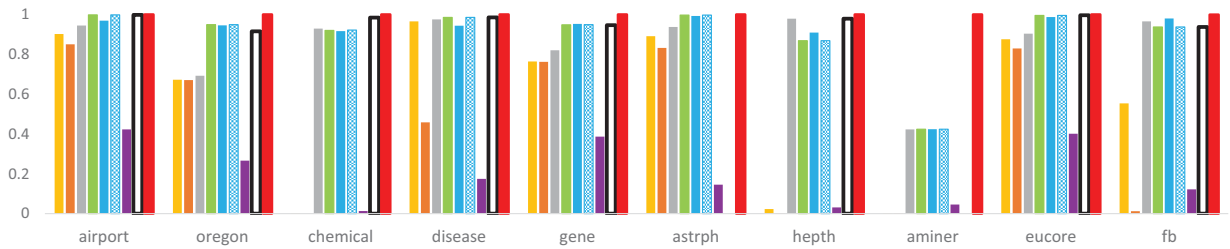
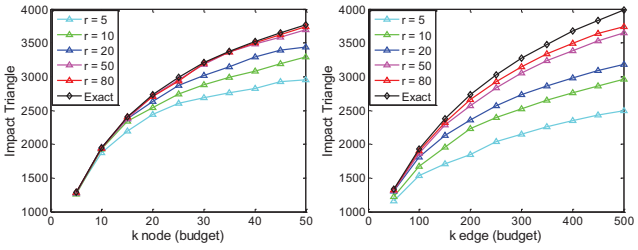
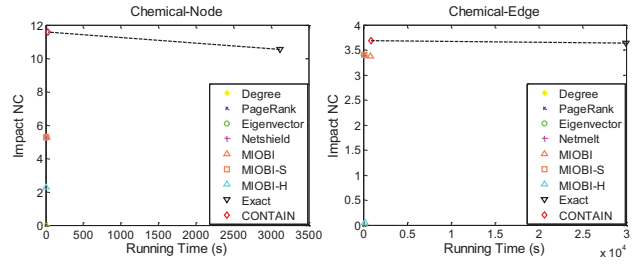


Figure 7: The optimization results on natural connectivity with edge-level operations.



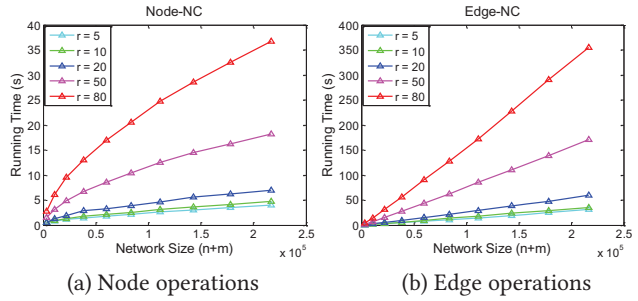
(a) Node operations. (b) Edge operations.

Figure 8: The effect of r on optimizing the number of triangles on CHEMICAL dataset.



(a) Node operations (b) Edge operations

Figure 9: The quality vs. running time trade-off on CHEMICAL. The budget for node operations is $k = 20$, the budget for edge operations is $k = 200$.



(a) Node operations (b) Edge operations

Figure 10: The scalability of CONTAIN. The budget for both node and edge operations is $k = 20$.

ACKNOWLEDGMENTS

This material is supported by the National Science Foundation under Grant No. IIS-1651203, IIS-1715385, IIS-1743040, ECCS-1547294, and CNS-1629888, by DTRA under the grant number HDTRA1-16-0017, by Army Research Office under the contract number W911NF-16-1-0168, by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001, and by additional gifts from Huawei and Baidu.

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] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 1999. Internet: Diameter of the world-wide web. *nature* 401, 6749 (1999), 130.

[2] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 2000. Error and attack tolerance of complex networks. *nature* 406, 6794 (2000), 378–382.

[3] Béla Bollobás. 2001. *The Evolution of Random Graphs–the Giant Component* (2 ed.). Cambridge University Press, 130–159.

[4] Deepayan Chakrabarti, Yang Wang, Chenxi Wang, Jurij Leskovec, and Christos Faloutsos. 2008. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security (TISSEC)* 10, 4 (2008), 1.

[5] Hau Chan, Leman Akoglu, and Hanghang Tong. 2014. Make it or break it: Manipulating robustness in large networks. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 325–333.

[6] Hau Chan, Shuchu Han, and Leman Akoglu. 2015. Where graph topology matters: the robust subgraph problem. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 10–18.

[7] Chen Chen, Jingrui He, Nadya Bliss, and Hanghang Tong. 2015. On the Connectivity of Multi-layered Networks: Models, Measures and Optimal Control. In *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 715–720.

[8] Chen Chen and Hanghang Tong. 2017. On the eigen-functions of dynamic graphs: Fast tracking and attribution algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 10, 2 (2017), 121–135.

[9] Chen Chen, Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. 2016. Eigen-Optimization on Large Graphs by Edge Manipulation. *ACM TKDD* 10, 4, Article 49 (2016), 30 pages.

[10] Chen Chen, Hanghang Tong, B Aditya Prakash, Charalampos E Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. 2016. Node Immunization on Large Graphs: Theory and Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2016), 113–126.

[11] Xilun Chen and K Selcuk Candan. 2014. LWI-SVD: low-rank, windowed, incremental singular value decompositions on time-evolving data sets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 987–996.

[12] Allan Peter Davis, Cynthia J Grondin, Kelley Lennon-Hopkins, Cynthia Saraceni-Richards, Daniela Sciaky, Benjamin L King, Thomas C Wiegers, and Carolyn J Mattingly. 2015. The Comparative Toxicogenomics Database's 10th year anniversary: update 2015. *Nucleic acids research* 43, D1 (2015), D914–D920.

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

[14] Mark Jerrum and Alistair Sinclair. 1988. 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, 235–244.

[15] Ling Jian, Jundong Li, and Huan Liu. 2018. Toward online node classification on streaming networks. *Data Min. Knowl. Discov.* 32, 1 (2018), 231–257.

[16] WU Jun, Mauricio Barahona, Tan Yue-Jin, and Deng Hong-Zhong. 2010. Natural connectivity of complex networks. *Chinese physics letters* 27, 7 (2010), 078902.

[17] Samir Khuller, Anna Moss, and Joseph Seffi Naor. 1999. The budgeted maximum coverage problem. *Inform. Process. Lett.* 70, 1 (1999), 39–45.

[18] Jon Kleinberg and Eva Tardos. 2006. *Algorithm design*. Pearson Education India.

[19] Istvan A Kovacs and Albert-Laszlo Barabasi. 2015. Network science: Destruction perfected. *Nature* 524, 7563 (2015), 38–39.

[20] Long T Le, Tina Eliassi-Rad, and Hanghang Tong. 2015. MET: A fast algorithm for minimizing propagation in large graphs with small eigen-gaps. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 694–702.

[21] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 177–187.

[22] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM TKDD* 1, 1 (2007), 2.

[23] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed Network Embedding for Learning in a Dynamic Environment. In *Proceedings of CIKM 2017*. ACM, 387–396.

[24] Liangyue Li, Hanghang Tong, Yanghua Xiao, and Wei Fan. 2015. Cheetah: fast graph kernel tracking on dynamic graphs. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 280–288.

[25] Rong-Hua Li and Jeffrey Xu Yu. 2015. Triangle minimization in large networks. *Knowledge and Information Systems* 45, 3 (2015), 617–643.

[26] Julian Mcauley and Jure Leskovec. 2014. Discovering social circles in ego networks. *ACM TKDD* 8, 1 (2014), 4.

[27] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

[28] Mark EJ Newman. 2008. The mathematics of networks. *The new palgrave encyclopedia of economics* 2 (2008), 1–12.

[29] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report.

[30] B Aditya Prakash, Deepayan Chakrabarti, Nicholas C Valler, Michalis Faloutsos, and Christos Faloutsos. 2012. Threshold conditions for arbitrary cascade models on arbitrary networks. *Knowledge and information systems* 33, 3 (2012), 549–575.

[31] Michael Sipser. 1997. *Introduction to the theory of computation.* PWS Publishing Company.
[32] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 990–998.
[33] Charalampos E Tsourakakis. 2008. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 608–617.
[34] Stanley Wasserman and Katherine Faust. 1994. *Social network analysis: Methods and applications*. Vol. 8. Cambridge university press.
[35] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. 2017. Local Higher-Order Graph Clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 555–564.
[36] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A Local Algorithm for Structure-Preserving Graph Cut. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 655–664.

8 APPENDIX: PROOF OF THEOREM 1

Proof. As NETCOP problem admits two possible network operations, including node deletions and edge deletions, we present our proof for each scenario in the following two lemmas. Lemma 5 and Lemma 6 would prove that NETCOP problem is NP-hard. □

Lemma 5. *The k-node connectivity minimization is NP-hard.*

Proof. By Eq. (1), the connectivity of network G is defined as $C(G) = \sum_{\pi \in G} f(\pi)$. We set function f as

$$f(\pi) = \begin{cases} 1 & \text{if } \pi \text{ is a valid subgraph} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

In other words, $C(G)$ is measured by the number of valid subgraphs in the network. Hence, we formulate the k-node minimization problem as follows.

Problem 2. k-Node Minimization Problem: $NodeMin(G, k)$
Given: *(1) A network G; (2) a set of non-independent valid subgraphs $\mathcal{S}_\pi$ and (3) a positive integer budget k with $1 < k < \min\{|\mathcal{S}_\pi|, |\mathcal{D}|\}$ where $\mathcal{D}$ is the set of nodes in G that incident to valid subgraphs.*
Output: *A set with k nodes, whose removal from G would minimize the number of valid subgraphs $|\mathcal{S}_\pi|$.*

Here we prove that $NodeMin(G, k)$ is NP-hard by constructing a polynomial reduction from a well-known NP-hard problem, the max k-hitting set problem ($MaxHit(n, m, k)$) [18]. The $MaxHit(n, m, k)$ problem is defined as follows.

Problem 3. Max k-Hitting Set Problem: $MaxHit(n, m, k)$
Given: *(1) a set $\mathcal{U}$ of n elements; (2) a collection $\mathcal{S} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m\}$ of m distinct subsets of $\mathcal{U}$, which are not mutually exclusive and (3) a positive integer k.*
Output: *A set $\mathcal{H} \subseteq \mathcal{U}$ with k elements, such that the cardinality of $\{\mathcal{B}_i | \mathcal{B}_i \cap \mathcal{H} \neq \phi\}$ is maximized.*

We aim to prove that $MaxHit(n, m, k)$ is polynomially reducible to $NodeMin(G, k)$ (i.e. $MaxHit(n, m, k) \leq_p NodeMin(G, k)$). Without loss of generality, we assume that $1 < k < \min\{n, m\}$. The rationality behind this assumption is that when $k = 1$, $MaxHit(n, m, 1)$ can be trivially solved by picking the element in $\mathcal{U}$ with the most associated sets from $\mathcal{S}$; when $k \geq m$, we can hit all m sets with at most m elements from $\mathcal{U}$, which is guaranteed by the pigeonhole principle; when $k \geq n$, the entire element set $\mathcal{U}$ could be picked, which returns a maximum possible solution for the given scenario.

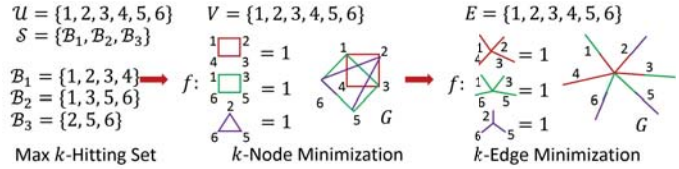


Figure 11: An illustration of polynomial reduction from Max k Hitting Set problem.

Given an instance of $MaxHit(n, m, k)$ with $1 < k < \min\{n, m\}$, we can construct a network G with n nodes, each corresponds to one element in $\mathcal{U}$. For each subset $\mathcal{B}_i \subseteq \mathcal{U}$, we construct a connected component G_i with $|\mathcal{B}_i|$ nodes in arbitrary shape as a valid subgraph. The vertices in G_i are the nodes corresponding to the elements in $\mathcal{B}_i$. Thus, the removal of any vertex in G_i would destroy the completeness of G_i. Consequently, the corresponding subgraph would become invalid. Therefore, we can construct G as the union of m valid subgraphs as $G = G_1 \cup G_2 \ldots \cup G_m$. Since the sets in $\mathcal{S}$ are distinct and not mutually exclusive, the resulting valid subgraph set $\mathcal{S}_\pi$ is guaranteed to be non-independent. Therefore, the solution of $MaxHit(n, m, k)$ would be equivalent to the solution of $NodeMin(G, k)$, which completes the proof. □

Figure 11 gives an illustration of the reduction from an instance of $MaxHit(n, m, k)$ to $NodeMin(G, k)$, in which valid subgraphs are marked with different colors. Edge $\langle 5, 6 \rangle$ has two colors because it participates in two different valid subgraphs.

Lemma 6. *The k-edge connectivity minimization is NP-hard.*

Proof. We also use the connectivity measure defined Eq. (15) to complete the proof. The corresponding k-edge minimization problem can be defined as follows.

Problem 4. k-Edge Minimization Problem ($EdgeMin(G, k)$)
Given: *(1) A network G; (2) a set of non-independent valid subgraphs $\mathcal{S}_\pi$ and (3) a positive integer budget k with $1 < k < \min\{|\mathcal{S}_\pi|, |\mathcal{D}|\}$ where $\mathcal{D}$ is the set of edges in G that were contained in valid subgraphs.*
Output: *A set with k edges, whose removal from G would minimize the number of valid subgraphs $|\mathcal{S}_\pi|$.*

We prove that $EdgeMin(G, k)$ is NP-hard by constructing a polynomial reduction from $MaxHit(n, m, k)$ problem. Similar to the rationale in the previous proof, we assume that $1 < k < \min\{n, m\}$.

Given an instance of $MaxHit(n, m, k)$, we construct a n-edge star-shaped network G (i.e. all the n edges share one common endpoint). Each edge corresponds to one element in $\mathcal{U}$. Given a subset $\mathcal{B}_i$, we first locate the corresponding edges in G based on the elements in $\mathcal{B}_i$, and then mark the sub-star formed by those edges as a valid subgraph G_i. Consequently, we have m valid subgraphs in G. The removal of any edge from G_i would destroy the completeness of the corresponding valid subgraph. Similarly, as the sets in $\mathcal{S}$ are distinct and not pair-wise mutually exclusive, the resulting valid subgraph set $\mathcal{S}_\pi$ is guaranteed to be non-independent. Therefore, the solution of $MaxHit(n, m, k)$ would be equivalent to the solution of $EdgeMin(G, k)$, which completes the proof. □

Figure 11 gives an illustration of the reduction from an instance of $MaxHit(n, m, k)$ to $EdgeMin(G, k)$. Again, edges with multiple colors indicate their participation in multiple valid subgraphs.